Journal of Artificial Intelligence Research 82 (2025) 1205-1242 Submitted 12/2024; published 03/2025

A Framework for Belief-based Programs and Their Verification

Daxin Liu DAXIN.LIU@NJU.EDU.CN
Nanjing University
Nanjing, Jiangsu 210023 China

Gerhard Lakemeyer GERHARD@KBSG.RWTH-AACHEN.DE
RWTH Aachen University
Aachen, NRW 52074 Germany

Abstract

Belief-based programming is a probabilistic extension of the GOLOG program family where every action and sensing result can be noisy and every test condition refers to the agent's subjective beliefs. Inherited from GOLOG programs, the action-centered feature makes belief programs fairly suitable for high-level robot control under uncertainty. An important step before deploying such a program is to verify whether it satisfies certain properties. At least two problems exist in verifying such programs: how to formally specify program properties and what is the complexity of the verification problem.

In this paper, we propose a formalism for belief programs based on a modal logic of actions and beliefs which allows us to conveniently express PCTL-like temporal properties. We also investigate the decidability and undecidability of the verification problem.

1. Introduction

Golog (Levesque et al., 1997) is a high-level logic programming language for the specification and execution of complex actions in dynamical domains based on the situation calculus (McCarthy, 1963; McCarthy & Hayes, 1981; Reiter, 2001). The idea behind Golog is to define powerful program constructs like *iteration, test, non-determinism*, and *(while) loops* as macros, which eventually expand into situation calculus formulas. The action-centered feature makes such a language rather suitable for robot control. Since its proposal, Golog has been expanded to incorporate features such as *coccurrency, interrupts, exogenous actions*, see ConGolog (Giacomo et al., 2000), *knowledge and sensing*, see IndiGolog (De Giacomo & Levesque, 1999; Sardiña et al., 2004), *decision theory*, see DTGolog (Boutilier et al., 2000; Soutchanski, 2001). More recently, Belle and Levesque (2015) proposed an extension called belief-based programs (or belief programs), where actions and sensing could be *noisy*. With the feature that test conditions refer to the agent's subjective *degree of belief*, belief programs are fairly suitable for robot control in a *partially observable* uncertain environment.

For safety and economic reasons, verifying such a program to ensure that it satisfies certain properties as designed before deployment is essential and desirable. As an illustrative example, consider a robot searching for coffee in a one-dimensional world as in Fig 1. Initially, the horizontal position *hpos* of the robot is at 0 and the coffee is at 2, the space is infinite, namely *hpos* could be any integer. Additionally, the robot has a knowledge base about its own location (usually a belief distribution). The robot might perform noisy

1 **while** $\boldsymbol{B}(hpos = 2) < 1$ **do**
2 $east(1)\|sensecoffee$;
3 **endWhile**

Table 1: An online belief program for the coffee robot.

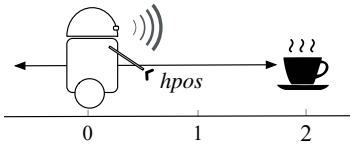

Figure 1: A coffee searching robot.

sensing *sensecoffee* to detect whether its current location has the coffee or not and an action $east(1)$ to move one unit east. The action $east(1)$ might be stochastic and end up moving, say, 1 or 2 units, causing the agent's belief about its position to shift to another distribution. Likewise, *sensecoffee* might be noisy, yet receiving a sensing result will somehow strengthen the agent's belief. A possible belief program is given in Table. 1. While the robot does not fully believe that it reached the coffee (Line 1), it non-deterministically selects the action $east(1)$ or sensing *sencfe* to execute (Line 2). The program is an online program and its execution depends on the outcome of sensing.

Some example properties one may be interested in are:

1. **P1**: whether the probability that within 2 steps of the program's execution the robot *believes* it reached the coffee with certainty is higher than 0.5;

2. **P2**: whether it is almost certain that *eventually* the robot *believes* it reached the coffee with certainty.

Often, the above program properties are specified by temporal formulas via Probabilistic Computational Tree Logic (PCTL) (Hansson & Jonsson, 1994). Obtaining the answer of whether a program satisfies the properties is non-trivial as the result depends on both the physical world, e.g. the robot's position and action models of actuators and sensors, and the robot's epistemic state, e.g. the robot's beliefs about its position and action models. There are at least two questions in verifying such belief programs: 1. how can we formally specify temporal properties like the ones above; 2. what is the computational complexity of the verification problem?

The semantics of belief programs proposed by Belle and Levesque (2015) is based on the seminal BHL formalism (Bacchus et al., 1999), which combines the *situation calculus* and probabilistic reasoning in a purely axiomatic fashion. While verification in this fashion has been studied for the non-probabilistic setting (Giacomo et al., 1997), such an axiomatic approach has some drawbacks. First, from the presentation point of view, the axiomatic approach tends to be cumbersome as it relies on the μ calculus where the use of second-order logic is inescapable. For instance, specifying the classical *request-and-response* property

(every request will eventually be responded to) is non-trivial:

$$(\forall x, \delta, s)Trans^*(\delta_0, S_0, \delta, do(request(x), s)) \supset ERes(x, \delta, do(request(x), s))$$

where $Trans^*$ refers to the transitive closure of the $Trans$ predicate (a predicate axiomatically defining the transitions among program configurations $\langle \delta, s \rangle$) and eventually responded $ERes$ is defined by

$$\begin{aligned} ERes(x, \delta_1, s_1) ::= \mu_{P,\delta,s}\{&[(\exists s'')s = do(selectRequest(x), s'')] \vee \\ &[((\exists \delta', s')Trans(\delta, s, \delta', s')) \wedge (\forall \delta', s')Trans(\delta, s, \delta', s') \supset P(\delta', s')]\}(\delta_1, s_1). \end{aligned}$$

Here $\mu_{P,\delta,s}$ denotes a least fixpoint according to the following formula:

$$(\forall \vec{x}\,)\{\mu_{P,\vec{y}}\Phi(P, \vec{y}\,)(\vec{x}\,) \equiv [(\forall P)[(\forall y)\Phi(P, \vec{y}\,) \supset P(\vec{y}\,)] \supset P(\vec{x}\,)]\}.$$

Such an example illustrates that formulating properties in this manner is involved and may be difficult to grasp. Second, closely in spirit, Lakemeyer and Levesque (2011) noticed that the axiomatic situation calculus substantially complicates answering questions that are not direct entailment problems such as "if *Theory1* entails *Formula1*, is it also the case that *Theory2* entails *Formula2*". Yet, such kind of problem lies at the heart of *progression* with an *action theory* (introduced later). Lastly, from the perspective of studying the complexity of the verification problem, starting with a second-order logic makes things undecidable immediately. Hence, one might wish to rule out this source of undecidability and start with a less expressive formalism.

To overcome these drawbacks, in this paper, we propose a new semantics for belief programs based on the logic $\mathcal{DS}_p$ (Liu & Feng, 2023), a modal version of the BHL logic with a possible-world semantics. We will show that by going modal and with proper constraints, we can avoid the use of second-order logic in expressing program properties. As a result, our modal formalism makes it easier than the axiomatic approach to express temporal properties like *eventually* and *globally* by using the usual modalities **F** and **G** in temporal logic. Another benefit of using modal logic, as observed by Lakemeyer and Levesque (2011), is that model-theoretic frameworks tend to lend themselves to more concise proofs.

Besides the new semantics of belief programs, we study the (un-)decidability of the verification problem. Specifically, Claßen *et al.* (2014) observed that many dimensions affect the complexity of the Golog program verification including the *underlying logic*, the *program constructs*, and the *domain specifications*, a.k.a *action theories*. Arguably, the dimension of domain specification is less well-studied. Hence, we study the boundary of decidability of the verification for belief programs from this dimension. We show that the result is strongly negative. However, we also investigate a case where the problem is decidable.

The rest of the paper is organized as follows. Section 2 introduces the logic $\mathcal{DS}_p$. Subsequently, we present the proposed semantics and specification of temporal properties for belief programs in Section 3. In Section 4, we study the boundary of decidability of the verification problem in a specific dimension: the *domain specification*. Section 5 considers a special case where the problem is decidable. In Section 6 and 7, we review related work and conclude.

2. Logical Foundations

In this section, we review the logic $\mathcal{DS}_p$, a first-order probabilistic modal logic with modalities for actions and degrees of belief. We will also briefly touch on the projection problem in the logic. The projection problem is common in dynamic settings, which is to decide what holds after a sequence of actions.

2.1 The Logic $\mathcal{DS}_p$

While the $\mathcal{DS}_p$ logic includes all the features of first-order logic, which allows fluents with an arbitrary number of arguments, we only use the nullary fragment of the logic $\mathcal{DS}_p$ plus a special unary fluent to express action likelihood. Here fluents refer to functions whose values may change as a result of actions. As observed by Belle and Levesque (2013, 2018), allowing fluents with arguments in degrees of belief might result in having a joint distribution over possible worlds of infinitely many, perhaps uncountably many, random variables, which is not handled in $\mathcal{DS}_p$. It is until recently that a continuous modal logic (Liu et al., 2023a), the logic $\mathcal{PS}$, was proposed to handle such generality, extending its static predecessor (Feng et al., 2023). However, it is unclear how to perform projection reasoning, a fundamental problem in reasoning about actions, in $\mathcal{PS}$ (we will come back to this problem in the next subsection). On the other hand, Liu and Feng (2023) provided a solution to the projection problem in $\mathcal{DS}_p$ for the nullary fluent fragment by progression. Hence, we focus on the the nullary fragment of the logic $\mathcal{DS}_p$.

$\mathcal{DS}_p$ is a many-sorted logic and there are two sorts: *number* and *action*. By number, we mean *algebraic real number* (Hardy & Wright, 1995) which includes irrational numbers such as $\sqrt{2}$.

2.1.1 The Language

The logic features a countable set of so-called *standard names* $\mathcal{N}$, which is isomorphic with a fixed universe of discourse. Roughly, this amounts to having an infinite domain closure axiom together with the unique name assumption (Levesque & Lakemeyer, 2001). $\mathcal{N} = \mathbb{D} \cup \mathcal{N}_A$ where $\mathbb{D}$ and $\mathcal{N}_A$ are number standard names and action standard names, respectively. The language includes a modal operator $\boldsymbol{B}$ for *degrees of belief* and features *fluent* and *rigid* functions. The interpretation of fluent functions varies as the result of actions, yet the interpretation of rigid functions is fixed. For simplicity, all action functions are rigid. Formally, the symbols of the language include equality "=" and

- standard names $\{n, n', \ldots\}$ and variables $\{x, y, \ldots\}$;
- *rigid function* symbols $+$, $\times$, *sensecoffee*(1), etc.;
- finitely many nullary *fluent functions* $\{h_1, h_2, \ldots h_k\}$.
- a special unary fluent l of sort number that takes an action as its argument and returns the action's likelihood. E.g. $l(\textit{sensecoffee}(1))$ returns how likely the sensing action receives a positive signal.[1]

1. We do not include the usual *Poss* for action preconditions. Impossible actions are treated as actions with 0 likelihood.

The logical connectives are $\{\wedge, \forall, \neg\}$. We treat $\{\vee, \exists, \equiv, \supset\}$ as the usual syntactic abbreviations. For simplicity, no predicates are considered. However, we will use "$\leq$" (defined on numbers) by assuming there is a rigid function to denote it.

Terms are the least set of expressions such that

1. every variable and standard name is a term;
2. if $t_1, \ldots t_k$ are terms and f is a k-ary function symbol, then $f(t_1, \ldots t_k)$ is a term.

A term is said to be *rigid* if it does not mention fluents. *Ground terms* are terms without variables. *Primitive terms* are terms of the form $f(n_1, \ldots, n_k)$, where f is a function symbol and n_i are number standard names. We denote the set of primitive terms as $\mathcal{P}_t$. Additionally, we assume $\mathcal{N}_A$ is just the set of action primitive terms.[2] For example, the sensing action *sensecoffee*(1), i.e. the sensor of the coffee robot reads a positive signal, is considered a standard action name. Furthermore, $\mathcal{Z}$ refers to the set of all finite sequences of action standard names, including the empty sequence $\langle\rangle$.

Atomic formulas are expressions of the form $t_1 = t_2$ for terms t_1, t_2. Arbitrary formulas are formed with the usual connectives $\neg, \wedge$, the quantifier $\forall$, and three modal operators $[t_a], \Box, \boldsymbol{B}$. The two *action modalities* $[t_a]\alpha$ and $\Box\alpha$ are read, respectively, as "α holds after action t_a," and "α holds after any sequence of actions. The *epistemic modality* $\boldsymbol{B}(\alpha\colon r)$ should be read as "α is believed with a probability r". We use $\boldsymbol{K}(\alpha)$ as abbreviation of $\boldsymbol{B}(\alpha\colon 1)$. α_t^x is the formula obtained by substituting all free occurrences of x in α by t.

A *sentence* is a formula without free variables. We use TRUE as an abbreviation for $\forall x(x = x)$, and FALSE for its negation. A formula with no $\Box$ or $[t_a]$ is called *static*. A formula with no $\boldsymbol{B}$ is called *objective*. A formula with no fluent, $\Box$ or $[t_a]$ outside $\boldsymbol{B}$ is called *subjective*. A formula with no $\boldsymbol{B}$, $\Box$, $[t_a]$, and l is called a *fluent formula*. A fluent formula without fluent functions is called a *rigid formula*.

2.1.2 The Semantics

The semantics is given in terms of possible worlds. Intuitively, a *world* determines what holds initially and after any actions, and the agent's beliefs are interpreted by *distributions* over possible worlds. More formally, A **world** $w : \mathcal{P}_t \times \mathcal{Z} \mapsto \mathcal{N}$ is a mapping from the primitive terms $\mathcal{P}_t$ and action sequences $\mathcal{Z}$ to standard names $\mathcal{N}$ of the right sort, satisfying *rigidity* and *arithmetical correctness*. That is:

- *Rigidity:* If $t \in \mathcal{P}_t$ is rigid, then for all worlds w and all $z, z' \in \mathcal{Z}$, $w[t, z] = w[t, z']$;
- *Arithmetical Correctness:* any arithmetical expression is rigid and has its standard interpretation. E.g. $w[1 + 1, z] = 2$ for all w, z.

We denote the set of all such worlds as $\mathcal{W}$. Given $w \in \mathcal{W}$, $z \in \mathcal{Z}$, and a ground term t, we define $|t|_w^z$ (the denotation for t given w, z) by:

1. If $t \in \mathcal{N}$, then $|t|_w^z = t$;

2. Alternatively, one could assume action standard names syntactically look like constants as well, our approach here is just an implicit way for the unique names assumption. Namely, $A(\vec{x})$ and $A'(\vec{x})$ are different actions, and $A(\vec{x})$ and $A(\vec{y})$ are the same action if and only if $\vec{x}$ and $\vec{y}$ are the same.

2. $|f(t_1, \ldots, t_k)|^z_w = w[f(|t_1|^z_w, \ldots, |t_k|^z_w), z]$.

Note that for rigid ground terms t, $|t|^z_w$ and $|t|^z_{w'}$ are identical for all $w, w' \in \mathcal{W}, z \in \mathcal{Z}$, in which case we write $|t|$ for short.

By a **distribution** d, we mean a mapping from $\mathcal{W}$ to $\mathbb{R}^{\geq 0}$ and an **epistemic state** e is any set of distributions. A comment here is that we do not require distribution d to be normalized, i.e. "$\sum_{w \in \mathcal{W}} d(w) = 1$", as $\mathcal{W}$ is uncountable and such a summation is hence not well-defined. We will come back to this issue later. Another point is that the idea to incorporate a set of distributions instead of a single distribution in the epistemic state derives from the philosophical stance that *de re* knowledge about degrees of belief should not be valid (Gabaldon & Lakemeyer, 2007). Namely, if epistemic states only contain single distributions, formulas such as $\exists x.\boldsymbol{K}(\boldsymbol{B}(\phi\colon x))$ would become valid, which seems counter-intuitive as it would mean that the agent has complete knowledge about the degree of belief for every sentence. Allowing for multiple distributions easily avoids that problem. By a *model*, we mean a triple (e, w, z) consisting of an epistemic state e, a possible world w, and a sequence of actions z.

Truth of objective sentences is defined as follows:

- $e, w, z \models t_1 = t_2$ iff $|t_1|^z_w$ and $|t_2|^z_w$ are identical;
- $e, w, z \models \neg\alpha$ iff $e, w, z \not\models \alpha$;
- $e, w, z \models \alpha \wedge \beta$ iff $e, w, z \models \alpha$ and $e, w, z \models \beta$;
- $e, w, z \models \forall x.\alpha$ iff $e, w, z \models \alpha^x_n$ for every standard name n of the right sort;
- $e, w, z \models [t_a]\alpha$ iff $e, w, z \cdot n \models \alpha$ and $n = |t_a|^z_w$;
- $e, w, z \models \Box\alpha$ iff $e, w, z \cdot z' \models \alpha$ for all $z' \in \mathcal{Z}$.

where $z \cdot z'$ denotes the concatenation of action sequences z and z'.

To account for stochastic actions, $\mathcal{DS}_p$ uses a notion called *observational indistinguishability* among actions (Bacchus et al., 1999). The idea is that instead of saying stochastic actions have non-deterministic effects, $\mathcal{DS}_p$ says stochastic actions have non-deterministic alternatives which are mutually observationally indistinguishable from the agent's perspective and each of which has a deterministic effect. In the coffee robot example, to express that action $east(1)$ might end up moving 1 or 2 units east non-deterministically, $\mathcal{DS}_p$ uses two actions $east(1, 1)$ and $east(1, 2)$, and they are interpreted in that the robot intends to move 1 unit east but nature may select 1 or 2 as outcomes.

More formally, we assume only two types of action symbols are used: stochastic actions and sensing. Additionally, we assume stochastic actions have at most 2 arguments and sensing has at most 1 argument. We remark that allowing only one controllable and one uncontrollable variable for stochastic actions (likewise for sensing) is not really a limitation as multiple arguments can always be encoded into a single argument by some encoding function. Nevertheless, this assumption will simplify the presentation a lot.

For stochastic action $a(x, y)$, x is called the *controllable* and *observable* argument while y is called the *uncontrollable* and *unobservable* argument. For example the argument 1

in $east(1, 2)$ is controllable and observable while the argument 2 is uncontrollable and unobservable. The argument x for sensing $sen(x)$ is *observable* yet *uncontrollable*, e.g. the argument 1 in $sensecoffee(1)$.

Of course, the outcome, i.e. the uncontrollable argument, of a stochastic action or sensing is determined by nature (the world). Nature will also determine how likely each outcome is by the special fluent $l(a)$. Besides, after observing the outcome, the agent will change its belief according to the likelihood of the outcome.

Hence, with the above conventions, we define *observational indistinguishability* among action sequences as follows:

Definition 1 (Observational Indistinguishability)**.** We define $z \approx z'$:

1. $\langle\rangle \approx z'$ iff $z' = \langle\rangle$;
2. $z \cdot r \approx z'$ iff $z' = z^* \cdot r$, $z \approx z^*$ and r is a sensing action;
3. $z \cdot a(x, y) \approx z'$ iff $z' = z^* \cdot a(x, y')$ for some y' and $z \approx z^*$.

Two action sequences z and z' are considered observationally indistinguishable to the agent if they are both empty sequences (Item 1); or their last actions are identical sensing action and their prefixes up to the last action are observationally indistinguishable (Item 2); or their last actions are the same except for the uncontrollable argument and their prefixes up to the last actions are observationally indistinguishable (item 3). For example, $east(1, 1) \cdot east(1, 2) \approx east(1, 2) \cdot east(1, 3)$.

Moreover, we define the likelihood of an action sequence in a world.

Definition 2 (Action Sequence Likelihood)**.** Let $w \in \mathcal{W}$, we define: $l^* \colon \mathcal{W} \times \mathcal{Z} \mapsto \mathbb{R}^{\geq 0}$ as

1. $l^*(w, \langle\rangle) = 1$;
2. $l^*(w, z \cdot a) = l^*(w, z) \cdot n$ where $n = w[l(a), z]$.

In words, the empty sequence has likelihood 1 and the likelihood of an action sequence $z \cdot a$ is just the likelihood of the z times the likelihood of a after action sequence z.

Since $\mathcal{W}$ is uncountable, to obtain a well-defined sum over uncountably many worlds, the following conditions are used for evaluating beliefs (Belle et al., 2016):

Definition 3 (Normalization)**.** We define $\textsc{Bnd}, \textsc{Eq}, \textsc{Norm}$ for any distribution d and any set of pairs $\mathcal{V} = \{(w_1, z_1), (w_2, z_2), \ldots\}$ as follows:

1. $\textsc{Bnd}(d, \mathcal{V}, n)$ iff $\neg\exists k, (w_1, z_1), \ldots, (w_k, z_k) \in \mathcal{V}$ such that $\sum_{i=1}^{k} d(w_i) \times l^*(w_i, z_i) > n$.
2. $\textsc{Eq}(d, \mathcal{V}, n)$ iff $\textsc{Bnd}(d, \mathcal{V}, n)$ and there is no $n' < n$ such that $\textsc{Bnd}(d, \mathcal{V}, n')$ holds.
3. for any $\mathcal{U} \subseteq \mathcal{V}$, $\textsc{Norm}(d, \mathcal{U}, \mathcal{V}, n)$ iff $\exists b \neq 0$ such that $\textsc{Eq}(d, \mathcal{U}, b \times n)$ and $\textsc{Eq}(d, \mathcal{V}, b)$.

Intuitively, $\textsc{Norm}(d, \mathcal{U}, \mathcal{V}, n)$ says the ratio of summed weights (under distribution d) of worlds in $\mathcal{U}$ to worlds in $\mathcal{V}$ is n. $\textsc{Eq}(d, \mathcal{V}, n)$ and $\textsc{Bnd}(d, \mathcal{V}, n)$ express respectively that the summed weights of worlds in $\mathcal{V}$ is n and is bounded by n. Technically, since $\mathcal{V}$ might be uncountable, the condition Eq and Bnd on d ensure only a countable subset of $\mathcal{V}$ (the

enumeration of (w_i, z_i) in the definition of BND) will receive non-zero weight in d, hence one can construct a discrete probability space over this countable subset of $\mathcal{V}$ via d. Namely, d is indeed a discrete probability distribution over possible worlds under the constraint of NORM (see (Belle et al., 2016) for a proof).

Given e, w, z and sentence α, let $\|\alpha\|_{e,w,z} := \{\langle w', z'\rangle \mid z' \approx z, e, w', z' \models \alpha\}$. Namely, $\|\alpha\|_{e,w,z}$ is the set of all pairs of worlds and actions that might result in α under the epistemic state e. For a distribution d, we abbreviate $\text{NORM}(d, \|\alpha\|_{\{d\},w,z}, \|\text{TRUE}\|_{\{d\},w,z}, n)$ as $\text{NORM}(d, \|\alpha\|_{\{d\},w,z}, n)$. Now, we are ready to give truth conditions for $\boldsymbol{B}$. Let r be a rigid term.

- $e, w, z \models \boldsymbol{B}(\alpha\colon r)$ iff for all $d \in e$, $\text{NORM}(d, \|\alpha\|_{\{d\},w,z}, \|r\|)$;

We freely use $\boldsymbol{B}(\alpha) \leq r$ (or $\boldsymbol{B}(\alpha) < r$) as formulas. They should be understood as syntactic abbreviations for $\exists x.\boldsymbol{B}(\alpha\colon x) \wedge x \leq r$ and analogous for "$<$".

For a sentence α, we write $e, w \models \alpha$ to mean $e, w, \langle\rangle \models \alpha$. When Σ is a set of sentences and α is a sentence, we write $\Sigma \models \alpha$ (read: Σ logically entails α) to mean that for all e and w, if $e, w \models \alpha'$ for every $\alpha' \in \Sigma$, then $e, w \models \alpha$. Satisfiability and validity are defined in the usual way. If α is an objective formula, we write $w \models \alpha$ instead of $e, w \models \alpha$. Similarly, we write $e \models \alpha$ instead of $e, w \models \alpha$ if α is subjective.

2.2 Basic Action Theories and Projection

The projection problem is one of the most important problems in dynamic settings, which asks what holds after actions. To address the projection problem, one needs to specify the dynamics of a domain first, namely, how actions change the world.

2.2.1 Basic Action Theories

In the situation calculus, the dynamic aspects of a domain are specified by a set of axioms, called the *basic action theory* (BAT) (Reiter, 2001). The logic $\mathcal{DS}_p$ uses a variant of BATs. Given a finite set of nullary fluents $\mathcal{H}$, a BAT Σ over $\mathcal{H}$ consists of the union of the following sets:

- Σ_0: a set of fluent sentences describing what holds initially, i.e. *initial state axiom*;
- Σ_{post}: a set of *successor state axioms* (SSAs), one for each fluent h in $\mathcal{H}$, of the form[3]

$$\Box[a]h = u \equiv \gamma_h(u, a) \vee h = u \wedge \neg\exists u'.\gamma_h(u', a) \tag{1}$$

  to characterize action effects, also providing a solution to the *frame problem* (Reiter, 2001); Here $\gamma_h(u, a)$ is a fluent formula with free variables u, a and is functional in u. Intuitively, the axiom says it is always the case that if $\gamma_h(u, a)$ holds, then the new value of h after executing a equals u, otherwise, the value of h remains unchanged.

- Σ_l: a set of likelihood axioms, one for each action symbol, of the form (for stochastic actions and sensing, respectively)

3. Free variables are implicitly universally quantified from the outside. The $\Box$ modality has lower syntactic precedence than the connectives, and $[\cdot]$ has the highest priority.

$$\Box l(a(x,y)) = u \equiv \phi_a(x,y,u)$$
$$\Box l(sen(x)) = u \equiv \phi_{sen}(x,u)$$

where $\phi_a(x,y,u)$ (likewise for $\phi_{sen}(x,u)$) is a fluent formula with free variables x, y, u (x,u) and is functional in u. Namely, it is always the case that the likelihood of stochastic action $a(x,y)$ equals u iff $\phi_a(x,y,u)$ holds (likewise for $sen(x)$).

Again, we do not include the usual *action precondition axioms*, and impossible actions are treated as actions with 0 likelihood. By a *belief distribution*, we mean the joint distribution of a finite set of random variables. Formally, assuming $\mathcal{H} = \{h_1, \ldots, h_m\}$, a belief distribution $\boldsymbol{B}^f$ of $\mathcal{H}$ is a formula of the form $\forall \vec{u}.\boldsymbol{B}(\vec{h} = \vec{u} : f(\vec{u}))$, where $\vec{u}$ is a set of variables, $\vec{h} = \vec{u}$ stands for $\bigwedge h_i = u_i$, and f is a rigid mathematical function of sort number with free variables $\vec{u}$.

Note that our choice of $\boldsymbol{B}^f$ essentially limits the epistemic state to a single distribution. This is needed because we rely on the projection mechanism by progression due to Liu and Feng (2023) which is only defined for single distributions. It is an interesting open question how to generalize their results to epistemic states with multiple distributions.

We will use $\Sigma_\Box$ to denote the union of Σ_{post} and Σ_l as they describe universal laws obeyed by the domain under consideration. Finally, by a knowledge base (KB), we mean a sentence of the form $\boldsymbol{B}^f \wedge \boldsymbol{K}\Sigma_\Box$. We require that the universal laws $\Sigma_\Box$ of a BAT Σ of the actual world is the same as the counterpart known by the agent. In case the two universal laws do not coincide, the physical world might select an outcome for sensing that the agent believes to be impossible, making the agent's belief inconsistent. This is also why the progression of belief (Theorem 2 below) is only defined for sensing which is known to have a non-zero likelihood by the agent.

Example 1. Consider the domain of the coffee robot in the introduction as follows. The only fluent of interest is the robot's position denoted by *hpos*. The position of the robot, i.e. *hpos*, can only be changed by actions $east(x,y)$; the changed value is determined by nature's choice y, not the intended value x; the exact distance moved is unobservable to the agent; for the stochastic action $east(x,y)$, with the half-half likelihood the exact distance y moved equals to x or $x+1$ where x is the intended value. Sensing $sensecoffee(x)$ is accurate, i.e. if the robot is at the location of coffee $hpos = 2$, a positive signal is read with likelihood 1, likewise if $hpos \neq 2$, a negative signal is read with certainty. Initially, the robot is at a certain non-positive position $hpos \leq 0$ and it fully believes its position is at 0 and knows the dynamics described above. These can be formally specified as below.

$$\Box[a]hpos = u \equiv \exists x, y.a = east(x,y) \wedge u = hpos + y$$
$$\vee\, hpos = u \wedge \neg\exists u'(\exists x, y.a = east(x,y) \wedge u' = hpos + y) \tag{2a}$$
$$\Box l(east(x,y)) = u \equiv u = 0.5 \wedge (y = x \vee y = x+1)$$
$$\vee\, u = 0 \wedge \neg(y = x \vee y = x+1) \tag{2b}$$
$$\Box l(sensecoffee(x)) = u \equiv u = 1 \wedge (x = 1 \wedge hpos = 2 \vee x = 0 \wedge hpos \neq 2)$$
$$\vee\, u = 0 \wedge (x = 1 \wedge hpos \neq 2 \vee x = 0 \wedge hpos = 2) \tag{2c}$$

Eq.(2a) says that after an action a, the robot's position is at u, if a is an $east(x,y)$ action and the new position u equals to the robot's original position *hpos* plus the moved

distance y, otherwise, the robot's position remains unchanged ($hpos = u$). Eq.(2b)-(2c) just specify what is described above. In addition, we have that $\Sigma_0 = \{hpos \leq 0\}$, and the agent's knowledge base is $\boldsymbol{B}^{f_0} \wedge \boldsymbol{K}\Sigma_\Box$, where $\Sigma_\Box$ is consists of Eq.(2a)-(2c), and $\boldsymbol{B}^{f_0} := \forall u.\boldsymbol{B}(hpos = u\colon f_0(u))$ is the initial belief distribution, here $f_0(u) := \mathit{if}\ u = 0\ \mathit{then}\ 1\ \mathit{otherwise}\ 0$. □

2.2.2 Projection by Progression

As mentioned before, projection in general is to decide what holds after actions. Progression is a solution to projection and the idea is to change the initial state according to the effects of actions and then evaluate queries against the updated state. Lin and Reiter (1997) showed that progression is second-order definable and conjectured that the use of second-order logic is inevitable which was confirmed later by Vassos and Levesque (2008). Recently, Liu and Feng (2023) showed that if all fluents are nullary, for the objective fragment, progression is first-order definable.

Theorem 1 (Liu and Feng (2023))**.** *Let $\Sigma := \Sigma_0 \cup \Sigma_\Box$ be a BAT, where Σ_0 is a set of fluent sentences (over the nullary fluents in $\mathcal{H}$), and t a ground action, then the following FO sentence is a progression of Σ_0 wrt $\Sigma_\Box$ and t:*

$$\exists \vec{v}.(\Sigma_0)^{\vec{h}}_{\vec{v}} \wedge \bigwedge_{h \in \mathcal{H}} \forall u.h = u \equiv (\phi_h)^{a,\vec{h}}_{t,\vec{v}} \tag{3}$$

where ϕ_h is the right hand side of the SSA in Eq.(1)*.*

For example, let $\Sigma_0 = \{hpos \leq 0\}$ and $\Sigma_\Box$ be as in Example 1, then the progression of Σ_0 wrt $\Sigma_\Box$ and action $east(1,2)$ is

$$\begin{aligned}\forall v.(v \leq 0) \wedge [hpos = u \equiv \exists x, y.east(1,2) = east(x,y) \wedge u = v + y \\ \vee\ v = u \wedge \neg\exists u'(\exists x, y.east(1,2) = east(x,y) \wedge u' = v + y)].\end{aligned}$$

After simplification, we have $\{hpos \leq 2\}$.

Let $Pro(\Sigma_0, \Sigma, t)$ be the FO progression of Σ_0 wrt Σ and action term t (or $Pro(\Sigma_0, t)$ for short). Liu and Feng also showed that the progression of a KB $\boldsymbol{B}^f \wedge \boldsymbol{K}\Sigma_\Box$ wrt to a stochastic action $t = a(x,y)$, denoted by $Pro(\boldsymbol{B}^f \wedge \boldsymbol{K}\Sigma_\Box, t)$, is another KB $\boldsymbol{B}^{f'} \wedge \boldsymbol{K}\Sigma_\Box$ with a belief distribution $\boldsymbol{B}^{f'}$:

Theorem 2 (Liu and Feng (2023))**.** *For all subjective α, $\boldsymbol{B}^f \wedge \boldsymbol{K}\Sigma_\Box \models [t]\alpha$ iff $\boldsymbol{B}^{f'} \wedge \boldsymbol{K}\Sigma_\Box \models \alpha$ where*

$$f'(\vec{u}) = \sum_{\vec{u}' \in \mathbb{D}^m} f(\vec{u}') \sum_{y' \in \mathbb{D}} l(a(x,y')) \times \mathbb{I}(\vec{u}, \vec{u}', a(x,y')) \tag{4}$$

and $\mathbb{I}$ is an indicator function given by $\mathbb{I}(\cdot) = 1$ if $Pro(\vec{h} = \vec{u}', a(x,y'))^{\vec{h}}_{\vec{u}}$ and 0 otherwise.

For sensing $sen(x)$ where $\boldsymbol{B}^f \wedge \boldsymbol{K}\Sigma_\Box \models \boldsymbol{K}(l(sen(x)) \neq 0)$, f' is given by $f'(\vec{u}) = \frac{1}{\eta} f(\vec{u}) \times l(sen(x))$, and η is a normalizer as $\eta = \sum_{\vec{u}' \in \mathbb{D}^m} f(\vec{u}') \times l(sen(x))$. In short, for a stochastic action t, the posterior f' is just the compound of the prior f and the likelihood distribution of t, while for sensing, the posterior f' is updated in a Bayesian way.

Example 2. Let $\boldsymbol{B}^{f_0} \wedge \boldsymbol{K}\Sigma_{\square}$ be as in Example 1, then its progression wrt the stochastic action $east(1,1)$ is $\boldsymbol{B}^{f_1} \wedge \boldsymbol{K}\Sigma_{\square}$, where f_1 is given by:

$$f_1(u) = \sum_{u' \in \mathbb{D}^m} f_0(u') \sum_{y' \in \mathbb{D}} l(east(1,y')) \times \mathbb{I}(u, u', east(1,y')) \tag{5a}$$

$$= f_0(0) \sum_{y' \in \mathbb{D}} \begin{cases} 0.5 & y' = 1 \vee y' = 1+1 \\ 0 & \neg(y' = 1 \vee y' = 1+1) \end{cases} \times \mathbb{I}(u, 0, east(1,y')) \tag{5b}$$

$$= \sum_{y' \in \mathbb{D}} \begin{cases} 0.5 & y' \in \{1,2\} \\ 0 & otherwise \end{cases} \times \begin{cases} 1 & [Pro(hpos = 0), east(1,y')]_u^{hpos} \\ 0 & otherwise \end{cases} \tag{5c}$$

$$= \sum_{y' \in \mathbb{D}} \begin{cases} 0.5 & y' \in \{1,2\} \\ 0 & otherwise \end{cases} \times \begin{cases} 1 & u = y' \\ 0 & otherwise \end{cases} \tag{5d}$$

$$= \begin{cases} 0.5 & u \in \{1,2\} \\ 0 & otherwise \end{cases} \tag{5e}$$

Here, Eq. (5a) holds by Theorem 2. Eq. (5b) holds because $f_0(u')$ is non-zero only if $u' = 0$. Eq. (5c) holds because $f'(0) = 1$ and by definition of $\mathbb{I}(\cdot)$. Eq. (5d) holds because the progression of $hpos = 0$ wrt $east(1,y')$ is $hpos = y'$, substituting $hpos$ with u thereafter leads to $u = y'$.

Likewise, the progression of $\boldsymbol{B}^{f_1} \wedge \boldsymbol{K}\Sigma_{\square}$ wrt the sensing action $sensecoffee(1)$ is $\boldsymbol{B}^{f_2} \wedge \boldsymbol{K}\Sigma_{\square}$ with

$$f_2(u) = \begin{cases} 1 & u = 2 \\ 0 & otherwise \end{cases}$$ □

We comment that the above result is a significant improvement and advancement over classical progression (Reiter, 2001) as the projection there cannot deal with belief and knowledge, it also goes beyond the epistemic non-probabilistic progression (Scherl & Levesque, 2003) or (Fang et al., 2015) since they cannot deal with stochastic actions. It goes beyond the progression result in the probabilistic situation calculus (Belle & Levesque, 2020) as it only works for so-called *invertible action theories*. Yet, there are also drawbacks as noted by Liu and Feng (2023). The progression result above makes use of infinite summation as a (rigid) logical term, which can be alternatively axiomatized by second-order logic. A problem in doing so is that summation is not closed under algebraic real numbers.[4] In other words, for some terms using infinite summation, a denotation may not exist. Another problem is that it is unknown whether it is decidable to check satisfiability wrt a theory involving summation.[5]

To sum up, for both theoretical and practical reasons, one may wish to avoid infinite summation. For this purpose, we impose some constraints on the basic action theories and

4. In fact, summation is not closed under *computable real number*. See also *Specker Sequence* in (Specker, 1949).
5. Since the Maclaurin expansion of the exponential function e^x can be expressed in terms of infinite summation, a positive answer to this would imply the theory of real with exponential is decidable. Yet, this problem has been open for decades (Macintyre et al., 1996).

knowledge base. Formally, we assume that Σ_l is of the form:

$$\begin{aligned}\Box l(a(x,y)) = u &\equiv \bigvee_{i,j} y = r_i(x) \wedge \phi_j(x) \wedge u = r_{i,j}(x)\\ \Box l(sen(x)) = u &\equiv \bigvee_{i,j} x = r'_i \wedge \phi_j \wedge u = r'_{i,j}\end{aligned} \tag{6}$$

where $r_i(x)$ and $r_{i,j}(x)$ are rigid terms with variables x; r'_i and $r'_{i,j}$ are rigid ground terms; $\phi_j(x)$, the *likelihood contexts*, are fluent formulas with free variables among x (likewise for sentence ϕ_j). Besides, we require that likelihood contexts are mutually exclusive and complete (likewise for sensing):

1. for distinct j and j', $\models \forall x.\phi_j(x) \supset \neg\phi_{j'}(x)$;
2. $\models \forall x. \bigvee_j \phi_j(x)$;
3. for all j, $\models \forall x. \sum_i r_{i,j}(x) = 1$.

The last item makes sure that for any x only finitely many outcomes of stochastic actions ($r_i(x)$) receive non-zero likelihoods ($r_{i,j}(x)$). A comment is that while the above assumption on the form in Eq. (6) is syntactic, the additional restrictions listed in the items above are semantic (by the use of "$\models$"). In general, it may be undecidable to check if an Σ_l satisfies these assumptions if one considers an expressive enough theory. Here, we leave it to the modeler to ensure that these assumptions are met.

In addition, we only consider belief distributions where only finitely many points receive non-zero degrees of belief. That is, $\forall \vec{u}.\boldsymbol{B}(\vec{h} = \vec{u} : f(\vec{u})) \equiv \bigwedge_i \boldsymbol{B}(\vec{h} = \vec{n}_i : r_i)$ for some vectors of standard name $\vec{n}_i$ and rigid ground terms r_i with $\sum_i r_i = 1$. Clearly, under these restrictions, the infinite summation in Eq. (4) can be converted into a formula with finite summation. As an example, the BAT in Example 1 satisfies the above restrictions.

3. The Proposed Framework

With a formalism of actions and degree of beliefs in hand, we can define belief-based programs, a type of probabilistic program where tests refer to the agent's subjective degree of belief.

3.1 Belief Programs

The atomic instructions of our belief programs are the so-called *primitive programs* which are actions that suppress their uncontrollable arguments. A primitive program ϱ can be *instantiated* by a ground action t_a, written $\varrho \to t_a$, if and only if $\models \exists y.\varrho[y] = t_a$, where $\varrho[y]$ is the action that restores its suppressed argument by y. For instance, the primitive program for the ground action $east(1,2)$ and $sensecoffee(1)$ are $east(1)$ and $sensecoffee$, i.e. $east(1) \to east(1,2)$, $sensecoffee \to sensecoffee(1)$, respectively.

Definition 4 (Program Expression)**.** A program expression δ is defined as:

$$\delta ::= \varrho \mid \alpha? \mid (\delta;\delta) \mid (\delta|\delta) \mid \delta^*$$

Namely, a program expression can be a primitive program ϱ, a test $\alpha?$ where α is a static subjective sentence, or constructed from sub-programs by sequence $\delta;\delta$, non-deterministic choice $\delta|\delta$, and non-deterministic iteration δ^*. Furthermore, **if** statements and **while** loops can be defined as abbreviations in terms of these constructs:

$$\textbf{if } \alpha \textbf{ then } \delta_1 \textbf{ else } \delta_2 \textbf{ endIf} := [\alpha?;\delta_1]|[\neg\alpha?;\delta_2]$$
$$\textbf{while } \alpha \textbf{ do } \delta \textbf{ endWhile} := [\alpha?;\delta]^*;\neg\alpha?$$

Under this convention, the coffee robot program δ_{cfe} in Table 1 just an abbreviation of

$$\delta_{cfe} := [\boldsymbol{B}(hpos = 2) < 1?;(east(1)|sensecoffee)]^*;\boldsymbol{B}(hpos = 2) = 1? \tag{7}$$

We remark that we disallow the so-called pick operator $\pi x.\delta$, which non-deterministically picks a program argument to execute. One reason to exclude it is that Claßen *et al.* (2014) proved that the pick operator is a source of undecidability for verification already in the non-probabilistic case.

Given a BAT $\Sigma := \Sigma_0 \cup \Sigma_\Box$, a knowledge base $\boldsymbol{B}^f \wedge \boldsymbol{K}\Sigma_\Box$, and a program expression δ, a **belief program** $\mathcal{BP}$ is a pair of the form

$$\mathcal{BP} = (\Sigma \wedge \boldsymbol{B}^f \wedge \boldsymbol{K}\Sigma_\Box, \delta).$$

For the coffee robot, a belief program could be

$$\mathcal{BP}_{cfe} = (h \leq 0 \wedge \Sigma_\Box \wedge \boldsymbol{B}^{f_0} \wedge \boldsymbol{K}\Sigma_\Box, \delta_{cfe}) \tag{8}$$

where $\Sigma_\Box$, $\boldsymbol{B}^{f_0}$ are as in Example 1 and δ_{cfe} as in Equation (7).

To give a uniform semantics for both terminating and non-terminating programs, we consider infinite execution paths of programs only and extend the finite execution paths of terminating programs with loops of trivial transitions (defined below). This is done by the following. First, to handle termination and failure, we reserve two nullary fluents $Final$ and $Fail$. Moreover,

$$\Box[a]Final = u \equiv a = \epsilon \wedge u = 1 \vee Final = u \wedge a \neq \epsilon$$
$$\Box[a]Fail = u \equiv a = \mathfrak{f} \wedge u = 1 \vee Fail = u \wedge a \neq \mathfrak{f}$$

is implicitly assumed to be part of $\Sigma_\Box$ where special actions $\{\epsilon, \mathfrak{f}\}$ are used to generate trivial transitions when programs terminate. Besides, we assume that $\Sigma_0 \models Final = 0 \wedge Fail = 0$, and actions $\epsilon, \mathfrak{f}$ do not occur in δ. Now, a *configuration* $\langle z, \delta\rangle$ consists of an action sequence z and a program expression δ. For a configuration $\langle z, \delta\rangle$, z represents the executed action history while δ represents the remaining programs expression to be executed.

Definition 5 (Program Semantics)**.** Let $\mathcal{BP} = (\Sigma \wedge \boldsymbol{B}^f \wedge \boldsymbol{K}\Sigma_\Box, \delta)$ be a belief program, the transition relation $\xrightarrow{e}$ among configurations, given an epistemic state e s.t. $e \models \boldsymbol{B}^f \wedge \boldsymbol{K}\Sigma_\Box$, is defined inductively:

1. $\langle z, \varrho\rangle \xrightarrow{e} \langle z \cdot t, \langle\rangle\rangle$, if $\varrho \to t$;
2. $\langle z, \delta_1;\delta_2\rangle \xrightarrow{e} \langle z \cdot t, \delta';\delta_2\rangle$, if $\langle z, \delta_1\rangle \xrightarrow{e} \langle z \cdot t, \delta'\rangle$;

3. $\langle z, \delta_1; \delta_2\rangle \xrightarrow{e} \langle z \cdot t, \delta'\rangle$, if $\langle z, \delta_1\rangle \in \mathrm{Fin}(e)$ and $\langle z, \delta_2\rangle \xrightarrow{e} \langle z \cdot t, \delta'\rangle$;
4. $\langle z, \delta_1|\delta_2\rangle \xrightarrow{e} \langle z \cdot t, \delta'\rangle$, if $\langle z, \delta_1\rangle \xrightarrow{e} \langle z \cdot t, \delta'\rangle$ or $\langle z, \delta_2\rangle \xrightarrow{e} \langle z \cdot t, \delta'\rangle$;
5. $\langle z, \delta^*\rangle \xrightarrow{e} \langle z \cdot t, \delta'; \delta^*\rangle$, if $\langle z, \delta\rangle \xrightarrow{e} \langle z \cdot t, \delta'\rangle$.

where $\mathrm{Fin}(e)$ is the set of *final configuration* wrt e given by:

1. $\langle z, \langle\rangle\rangle \in \mathrm{Fin}(e)$;
2. $\langle z, \alpha?\rangle \in \mathrm{Fin}(e)$ if $e, w, z \models \alpha$;
3. $\langle z, \delta_1; \delta_2\rangle \in \mathrm{Fin}(e)$ if $\langle z, \delta_1\rangle \in \mathrm{Fin}(e)$ and $\langle z, \delta_2\rangle \in \mathrm{Fin}(e)$;
4. $\langle z, \delta_1|\delta_2\rangle \in \mathrm{Fin}(e)$ if $\langle z, \delta_1\rangle \in \mathrm{Fin}(e)$ or $\langle z, \delta_2\rangle \in \mathrm{Fin}(e)$;
5. $\langle z, \delta^*\rangle \in \mathrm{Fin}(e)$;

The set of *failing configurations* is given by:

$$\mathrm{Fail}(e) = \{\langle z, \delta\rangle \mid \langle z, \delta\rangle \notin \mathrm{Fin}(e), \text{ there is no } \langle z \cdot t, \delta'\rangle \text{ s.t. } \langle z, \delta\rangle \xrightarrow{e} \langle z \cdot t, \delta'\rangle\}.$$

We extend final and failing configurations with additional *trivial transitions* by actions $\{\epsilon, \mathfrak{f}\}$. This means that all execution paths of the programs will now be infinite. We achieve this by defining an extension of $\xrightarrow{e}$. The *extended transition relation* $\overset{e}{\hookrightarrow}$ *among configurations* is defined as the least set such that:

1. $\langle z, \delta\rangle \overset{e}{\hookrightarrow} \langle z \cdot t, \delta'\rangle$ if $\langle z, \delta\rangle \xrightarrow{e} \langle z \cdot t, \delta'\rangle$;
2. $\langle z, \delta\rangle \overset{e}{\hookrightarrow} \langle z \cdot \epsilon, \langle\rangle\rangle$ if $\langle z, \delta\rangle \in \mathrm{Fin}(e)$;
3. $\langle z, \delta\rangle \overset{e}{\hookrightarrow} \langle z \cdot \mathfrak{f}, \langle\rangle\rangle$ if $\langle z, \delta\rangle \in \mathrm{Fail}(e)$.

The execution of a program $\mathcal{BP}$ yields a countably infinite[6] *Markov decision process* (MDP) $\mathsf{M}_\delta^{e,w}$ wrt e, w s.t. $e, w \models \Sigma \wedge \boldsymbol{B}^f \wedge \boldsymbol{K}\Sigma_\Box$.

Definition 6 (MDP)**.** A Markov decision process is a tuple $\mathsf{M} = (\mathsf{S}, \mathsf{Act}, \mathsf{P}, s_0)$ with a countable set S of states, an initial state s_0, a finite set Act of actions,[7] and a transition function $\mathsf{P} : \mathsf{S} \times \mathsf{Act} \times \mathsf{S} \mapsto [0, 1]$, with $\sum_{s' \in \mathsf{S}} \mathsf{P}(s, a, s) = 1$ for all $s \in \mathsf{S}$ and $a \in \mathsf{Act}$.[8]

In our program semantics, the MDP $\mathsf{M}_\delta^{e,w} = (\mathsf{S}, \mathsf{Act}, \mathsf{P}, s_0)$ is given by:

1. S : the set of configurations reachable from $\langle\langle\rangle, \delta\rangle$ under $\overset{e}{\hookrightarrow}^*$ (transitive and reflexive closure of $\overset{e}{\hookrightarrow}$);
2. Act : the finite set of primitive programs in δ;

6. Our restrictions on Σ_l ensure a bounded branching for the MDP, therefore its states are countable.
7. The notion of actions here is different from the notion of actions in the logic $\mathcal{DS}_p$ which are logical terms.
8. We leave out the classical reward function in MDP as we do not need it.

3. P : the transition function with $\mathsf{P}(\langle z,\delta\rangle, \varrho, \langle z\cdot t,\delta'\rangle)$ given by:

$$\mathsf{P}(\cdot)=\begin{cases} p & \varrho\to t\ ,\ w,z\models l(t)=p,\\ & \text{and } \langle z,\delta\rangle \overset{e}{\hookrightarrow} \langle z\cdot t,\delta'\rangle \\ 1 & \langle z,\delta\rangle\in \mathrm{Fin}(e) \text{ and } \varrho=t=\delta'=\epsilon \\ 1 & \langle z,\delta\rangle\in \mathrm{Fail}(e) \text{ and } \varrho=t=\mathfrak{f},\delta'=\delta \\ 0 & \textit{otherwise}\end{cases}$$

4. s_0: the initial state $\langle\langle\rangle,\delta\rangle$.

Now, the non-determinism on the agent's side is resolved by means of *policy* σ, which is a mapping $\sigma : \mathsf{S}\mapsto \mathsf{Act}$. A policy σ is said to be *proper* if and only if for all $s=\langle z,\delta\rangle$, $s'=\langle z',\delta\rangle$, if $z\approx z'$, then $\sigma(s)=\sigma(s')$, namely, the robot acts only according to its observed action histories (not the actual ones). A comment is that under our assumption of stochastic actions and sensing, $z\approx z'$ implies that $Pro(\boldsymbol{B}^f\wedge \boldsymbol{K}\Sigma_\Box, z)$ and $Pro(\boldsymbol{B}^f\wedge \boldsymbol{K}\Sigma_\Box, z')$ are equivalent given any initial KB $\boldsymbol{B}^f\wedge\boldsymbol{K}\Sigma_\Box$. Hence, proper policies also mean that the robot acts only according to its knowledge base KB.

An infinite path $\pi = s_0 \xrightarrow{\varrho_1} s_1 \xrightarrow{\varrho_2} s_2\cdots$ is called a *σ-path* if $\sigma(s_j)=\varrho_j$ for all $j\geq 0$. The j-th state of any such path is denoted by $\pi[j]$. The set of all *σ-paths* starting in s is denoted by $\mathsf{Path}^\sigma(s,\mathsf{M}_\delta^{e,w})$.

Every policy σ induces a probability space Pr_s^σ on the set of infinite paths starting in s, using the *cylinder set* construction (Kemeny, Snell, & Knapp, 2012): For any set of infinite paths starting with the finite path prefix $\pi_{\mathsf{fin}} = s_0 \xrightarrow{\varrho_1} s_1\cdots s_n$, we define the probability measure of the set of infinite paths as:

$$\mathsf{Pr}^\sigma_{s_0,\mathsf{fin}} = \mathsf{P}(s_0,\varrho_1,s_1)\cdot\mathsf{P}(s_1,\varrho_2,s_2)\cdot\ldots\cdot\mathsf{P}(s_{n-1},\varrho_n,s_n)$$

3.2 Temporal Properties of Programs

We use a variant of PCTL to specify program properties. The syntax is given as:

$$\Phi ::= \beta \mid \neg\Phi \mid \Phi\wedge\Phi \mid \mathbf{P}_I[\Psi] \tag{9a}$$

$$\Psi ::= \mathbf{X}\Phi \mid (\Phi\mathbf{U}\Phi) \mid (\Phi\mathbf{U}^{\leq k}\Phi) \tag{9b}$$

where β is any static subjective $\mathcal{DS}_p$ formula. We call formulas according to Eq. (9a) *state formulas* and according to Eq. (9b) *trace formulas*. Here $I\subseteq[0,1]$ is an interval. $\Phi\mathbf{U}^{\leq k}\Phi$ is the step-bounded version of the until operator. Some useful abbreviations are: $\mathbf{F}\Phi$ (eventually Φ) for $\textsc{true}\mathbf{U}\Phi$ and $\mathbf{G}\Phi$ (globally Φ) for $\neg\mathbf{F}\neg\Phi$.

Let Φ be a temporal state formula, Ψ a temporal trace formula, $\mathsf{M}_\delta^{e,w}$ the infinite-state MDP of a program $\mathcal{BP}=(\Sigma\wedge\boldsymbol{B}^f\wedge\boldsymbol{K}\Sigma_\Box,\delta)$ wrt e,w s.t. $e,w\models\Sigma\wedge\boldsymbol{B}^f\wedge\boldsymbol{K}\Sigma_\Box$, and $s\in\mathsf{S}$. The truth conditions of state formulas are given as:

1. $\mathsf{M}_\delta^{e,w}, s\models\beta$ iff $s=\langle z,\delta\rangle$ and $e,w,z\models\beta$;
2. $\mathsf{M}_\delta^{e,w}, s\models\neg\Phi$ iff $\mathsf{M}_\delta^{e,w}, s\not\models\Phi$;
3. $\mathsf{M}_\delta^{e,w}, s\models\Phi_1\wedge\Phi_2$ iff $\mathsf{M}_\delta^{e,w}, s\models\Phi_1$ and $\mathsf{M}_\delta^{e,w}, s\models\Phi_2$;

4. $\mathsf{M}_\delta^{e,w}, s \models \mathbf{P}_I[\Psi]$ iff for all *proper* policies σ, $\mathsf{Pr}_s^\sigma(\Psi) \in I$, where

$$\mathsf{Pr}_s^\sigma(\Psi) = \mathsf{Pr}_s^\sigma(\{\pi \in \mathsf{Path}^\sigma(s, \mathsf{M}_\delta^{e,w}) \mid \mathsf{M}_\delta^{e,w}, \pi \models \Psi\}).$$

Items 1-3 are trivial. Item 4 intuitively says that, $\mathsf{M}_\delta^{e,w}, s \models \mathbf{P}_I[\Psi]$ if and only for all the proper policies σ, the set of all infinite paths starting from s, that is induced by σ and satisfy the trace formula Ψ (defined below), has probabilities in terms of the probability space Pr_s^σ that lie in the interval I.

Furthermore, let $\pi \in \mathsf{Path}^\sigma(s, \mathsf{M}_\delta^{e,w})$ be an infinite path for some proper policy σ. Then the truth conditions of trace formulas are given as:

1. $\mathsf{M}_\delta^{e,w}, \pi \models \mathbf{X}\Phi$ iff $\mathsf{M}_\delta^{e,w}, \pi[1] \models \Phi$;
2. $\mathsf{M}_\delta^{e,w}, \pi \models \Phi_1 \mathbf{U} \Phi_2$ iff $\exists i. 0 \leq i$ s.t. $\mathsf{M}_\delta^{e,w}, \pi[i] \models \Phi_2$ and $\forall j. 0 \leq j \leq i, \mathsf{M}_\delta^{e,w}, \pi[j] \models \Phi_1$;
3. $\mathsf{M}_\delta^{e,w}, \pi \models \Phi_1 \mathbf{U}^{\leq k} \Phi_2$ iff $\exists i. 0 \leq i \leq k$ s.t. $\mathsf{M}_\delta^{e,w}, \pi[i] \models \Phi_2$ and $\forall j. 0 \leq j \leq i, \mathsf{M}_\delta^{e,w}, \pi[j] \models \Phi_1$;

In words: π (under $\mathsf{M}_\delta^{e,w}$) satisfies $\mathbf{X}\Phi$ if and only if the next state of path π satisfies state formula Φ; $\Phi_1 \mathbf{U} \Phi_2$ is satisfied by π if and only if along all the states of π, Φ_1 holds until Φ_2 holds; $\Phi_1 \mathbf{U}^{\leq k} \Phi_2$ is satisfied by π if and only if along all the states of π, Φ_2 will hold within at most k steps, where Φ_1 holds in all states before.

Definition 7 (The Verification Problem)**.** A temporal state formula Φ is valid in a program $\mathcal{BP}$, $\mathcal{BP} \models \Phi$, iff for all e, w with $e, w \models \Sigma \wedge \boldsymbol{B}^f \wedge \boldsymbol{K}\Sigma_\Box$, it holds that $\mathsf{M}_\delta^{e,w}, s_0 \models \Phi$. Given a program $\mathcal{BP}$ and a temporal state formula Φ, the verification problem is then to decide if $\mathcal{BP} \models \Phi$.

E.g. formulas $\mathbf{P}_{\geq 0.5}[\mathbf{F}^{\leq 2}\boldsymbol{B}(\mathit{hpos} = 2\colon 1)]$ and $\mathbf{P}_{=1}[\mathbf{F}\boldsymbol{B}(\mathit{hpos} = 2\colon 1)]$ specify the two properties **P1** and **P2** in the introduction respectively. The two properties are not satisfied by the coffee program $\mathcal{BP}_{cfe}$ in Eq. (8). We show that $\mathcal{BP}_{cfe} \not\models \mathbf{P}_{=1}[\mathbf{F}\boldsymbol{B}(\mathit{hpos} = 2\colon 1)]$, the other one is a direct consequence of this. Consider the world w where the robot is at $\mathit{hpos} = 0$, which satisfies the constraint $\mathit{hpos} \leq 0$ in Σ_0, in such a world, a policy σ of the robot program is to always select *sensecoffee* and never choose *east(1)*. Under the world w and policy σ, sensing will always end up receiving a negative signal and the robot's beliefs remain unchanged. Hence, $\boldsymbol{B}(\mathit{hpos} = 2\colon 1)$ will never be satisfied. By Definition 7, $\mathcal{BP}_{cfe} \not\models \mathbf{P}_{=1}[\mathbf{F}\boldsymbol{B}(\mathit{hpos} = 2\colon 1)]$.

We comment that formulas in the above examples contain two different types of modalities for probability $\mathbf{P}$ and $\boldsymbol{B}$. The probability space of $\boldsymbol{B}$ is defined over possible worlds, while the probability space of $\mathbf{P}$ is defined over the cylinder set of infinite paths. In some sense, $\boldsymbol{B}$ is subjective since the epistemic state e alone determines the truth value of a $\boldsymbol{B}$ formula. On the other hand, $\mathbf{P}$ is objective and the transition probability from one configuration to another is determined by the actual world w, hence $\mathbf{P}$ reflects a property of the physical world. One may be interested in whether such two kinds of probability can be unified into one, i.e. for any formula containing both $\boldsymbol{B}$ and $\mathbf{P}$, is there another formula with only $\boldsymbol{B}$ or $\mathbf{P}$ that is logically equivalent to it? This is answered negatively in the paper (Beauquier et al., 2006) as $\mathbf{P}$ depends on the branching of the underlying transition system while $\boldsymbol{B}$ does not. That is the two types of probability are indeed incompatible.

Another comment is that the semantics of **P** only considers proper policies that are determined by the agent's knowledge base. In other words, we are interested in the agent's strategies to achieve some state in its perspective, which is why β in Eq. (9a) is subjective. Alternatively, one could consider the set of all policies and use objective $\mathcal{DS}_p$ formulas in the state formula, in which case the focus is on whether the "physical world" has strategies to achieve some goals.

Finally, we remark that our approach does not support strategy synthesis. For example, questions such as whether there exists a strategy for executing a belief-based program such that a temporal property is achieved with a certain probability are not addressed. In contrast, we are concerned with the question of whether such properties hold for all proper strategies compatible with a belief-based program.

4. Undecidability

The verification problem is undecidable because belief programs are probabilistic variants of Golog programs with sensing, for which undecidability was shown by Zarrieß and Claßen (2016). Claßen *et al.* (2014) observed that many dimensions affect the complexity of the Golog program verification including the *underlying logic*, the *program constructs*, and the *domain specifications*. Since then, efforts have been made to find decidable fragments. Arguably, the dimension of domain specification is less well-studied. Here we study the boundary of decidability from this dimension. Hence, in this paper, we set the other two dimensions to a known decidable status.

Formally, we assume our logic only contains $+, \times$ as rigid function symbols, and whenever we write logical entailment $\Sigma \models \alpha$, we mean $\Sigma \cup \mathcal{T}_{\mathcal{R}} \models \alpha$ where $\mathcal{T}_{\mathcal{R}}$ is the theory of the reals, where validity is decidable (Tarski, 1998).

In deterministic settings, domain specifications mainly refer to SSAs. Nevertheless, in our case, the likelihood axiom plays an important role as well. Some relevant variants of SSAs are *strongly context-free* (Reiter, 2001) and *local-effect* SSAs (Liu & Levesque, 2005). Recall that our SSAs are of the form $\Box[a]h = u \equiv \gamma_h(u, a) \vee h = u \wedge \neg\exists u'.\gamma_h(u', a)$ for fluent h.

Definition 8. A set of SSAs is called:

1. strongly context-free, if for all fluents h, γ_h is a disjunction of formulas of the form $\exists\vec{\mu}.a = A(\vec{v})$, where A is an action symbol, $\vec{v}$ contains u and μ;

2. local-effect, if for all fluents h, γ_h is a disjunction of the form $\exists\vec{\mu}.a = A(\vec{v}) \wedge \nabla(\vec{v})$, where A is an action symbol, $\vec{v}$ contains u and μ, and $\nabla(\vec{v})$ is a fluent formula with free variables in $\vec{v}$.

$\nabla(\vec{v})$ is called the *effect condition* or *effect context*. For a local-effect SSA, instantiating $\gamma_h(u, a)$ wrt a ground action $A(\vec{t})$ results in a formula of the form $\bigvee u = t_i \wedge \nabla(t_j)$ where $t_i, t_j \in \vec{t}$ (The same holds for strongly context-free SSA but replacing $\nabla(t_j)$ by TRUE). That is, the new value of the fluent is determined by the action's arguments conditioned on the context formulas $\nabla(t_j)$. Strongly context-free SSAs are special local-effect SSAs where the context is vacuously true, i.e. the effects of an action are independent of the status of the

world. For example, the SSA

$$\Box[a]powerOn = x \equiv a = turnOn \wedge x = 1 \vee \\ powerOn = x \wedge a \neq turnOn$$

is strongly context-free, which says the ground action $turnOn$, irrespective of the current status of the world, will turn the power on ($powerOn = 1$). On the other hand, the SSA

$$\Box[a]level = x \equiv a = setLevel(x) \wedge powerOn = 1 \vee \\ level = x \wedge (\forall y.a \neq setLevel(y) \vee powerOn \neq 1)$$

is local-effect, which says that the action $setLevel(x)$ will set the level of a lift to x if the power is on, i.e. $powerOn = 1$. The SSA in Example 1 is not local-effect, as the new value of location h is not part of the argument $east(x, y)$.

We remark that the reason why we are interested in these two types of SSAs is that verifying CTL* properties for non-probabilistic Golog programs with local-effect SSAs was proven to be decidable (Zarrieß & Claßen, 2014). So restricting to local-effect SSAs for the verification of belief programs seems like a good starting point. Another reason is that starting with a finite belief distribution $\boldsymbol{B}^f$, for programs with local-effect SSA, belief distributions generated during execution will always be finite (hence avoid infinite summation). This is because (1) new values of fluents are given as action arguments; (2) every primitive program has finitely many instantiations of ground actions; (3) and there are finitely many primitive programs in a belief program. This might not be the case for non-local-effect SSAs. For our coffee robot's BAT in Example 1, if we iterate the primitive program $east(1)$ infinitely often, the possible locations of the robot may be arbitrary non-negative integers, hence the belief distribution is ultimately infinite.

Recall that our likelihood axioms for stochastic actions and sensing are of the form

$$\Box l(a(x, y)) = u \equiv \bigvee_{i,j} y = r_i(x) \wedge \phi_j(x) \wedge u = r_{i,j}(x)$$
$$\Box l(sen(x)) = u \equiv \bigvee_{i,j} x = r'_i \wedge \phi_j \wedge u = r'_{i,j}$$

Definition 9. A likelihood axiom is called strongly context-free if $\phi_j(x)$ and ϕ_j as above are equivalent to true.

Intuitively, for strongly context-free likelihood axiom, actions' likelihood is independent of the context. We comment that the strongly context-free likelihood axiom excludes sensing as sensing always involves fluents. It makes no sense to have a sensor that randomly reads a value. In fact, calling such "sensing" a stochastic action with no effects seems better.

Table 2 lists the undecidability of the belief program verification problem. Dashes mean no constraint. The result is arranged as follows. We first explore decidability for the case with no restriction on the likelihood axioms. As it turns out, the problem is undecidable even if SSAs are strongly context-free (1). Therefore, we set the likelihood axiom to be strongly context-free, which results in undecidability for the case of local-effect SSAs (2). The case with the question mark remains open (3).

#	likelihood axiom	SSA	Decidable
1	-	strongly context-free	No
2	strongly context-free	local-effect	No
3	strongly context-free	strongly context-free	?

Table 2: Decidability of the verification problem

Theorem 3. *The verification problem is undecidable for programs with strongly context-free SSAs.*

The proof is by a reduction to the undecidable *emptiness problem* of *probabilistic automata* (PA) (Paz, 2014). The idea is that for any given PA $\mathcal{A}$, we can construct a belief program $\mathcal{BP}$ to simulate its run. The language recognized by $\mathcal{A}$ with probability no less than a threshold $\xi \in [0,1]$ is empty if and only if $\mathcal{BP} \models \mathbf{P}_{=1}[\mathbf{G}\boldsymbol{B}(h_s \in \mathcal{N}_{\mathsf{F}}) < \xi]$, here fluent h_s records the current state of $\mathcal{A}$, $\mathcal{N}_{\mathsf{F}}$ is a finite set of standard names representing the accepting states of $\mathcal{A}$. Since the former is known to be undecidable, the verification problem is undecidable.

More formally, a PA is a quintuple $\mathcal{A} = (\mathsf{Q}, \mathsf{L}, (\mathsf{M}_l)_{l\in\mathsf{L}}, q_1, \mathsf{F})$ where Q is a finite set of states, L is a finite alphabet of letters, $(\mathsf{M}_l)_{l\in\mathsf{L}}$ are the stochastic transition matrices, $q_1 \in \mathsf{Q}$ is the initial state and $\mathsf{F} \subseteq \mathsf{Q}$ is a set of accepting states. For each letter $l \in \mathsf{L}$, $\mathsf{M}_l \in [0,1]^{\mathsf{Q}\times\mathsf{Q}}$ defines transition probabilities: $0 \le \mathsf{M}_l(q_i, q_j) \le 1$ is the transition probability from state q_i to q_j when reading a letter l. Given a probability distribution o over Q, and a letter $l \in \mathsf{L}$, let $o\cdot l$ be the probability distribution of states after l, i.e. $(o\cdot l)(q_j) = \sum_{q_i\in\mathsf{Q}} o(q_i)\cdot\mathsf{M}_l(q_i, q_j)$. This naturally extends to a sequence of L^* (namely word $\mathbf{w}$): $\forall \mathbf{w} \in \mathsf{L}^*, \forall l \in \mathsf{L}, o\cdot(\mathbf{w}l) = (o\cdot\mathbf{w})\cdot l$. For every state $q \in \mathsf{Q}$ and for any set of states $\mathsf{R} \subseteq \mathsf{Q}$, we denote $\mathbb{P}_{\mathcal{A}}(q \xrightarrow{\mathbf{w}} \mathsf{R}) = \sum_{q'\in\mathsf{R}} (o_q \cdot \mathbf{w})(q')$ as the probability to reach the set R from state q when reading word $\mathbf{w}$. The *emptiness problem* is that given a PA $\mathcal{A}$ and $\xi \in [0,1]$, deciding whether there exists a word $\mathbf{w}$ (a sequence of letters) such that $\mathbb{P}_{\mathcal{A}}(q_1 \xrightarrow{\mathbf{w}} \mathsf{F}) \ge \xi$, namely, the probability of reaching accepting states from the initial state upon reading $\mathbf{w}$ is no less than ξ. The emptiness problem is known to be undecidable.

The idea of the reduction is to use a strongly context-free stochastic action $\varrho_l(y)$ to simulate reading letter l. $\varrho_l(y)$'s effect is to set the current state h_s to y with a probability $\mathsf{M}_l(h_s, y)$, (just like the transition probability $\mathsf{M}_l(q_i, q_j)$). Moreover, the constructed belief program simply iteratively non-deterministically selects a primitive program ϱ_l to execute until $\boldsymbol{B}(h_s \in \mathcal{N}_{\mathsf{F}}) \ge \xi$ holds. The simulation is sound if a distribution o over Q coincides with a belief distribution $\boldsymbol{B}^f$ of h_s, i.e. for all $q_i \in \mathsf{Q}$, $o(q_i) = f(n_i)$ where n_i is the standard name representing the state q_i in $\mathcal{A}$, then for all $n_j \in \mathsf{Q}$, $(o\cdot l)(q_j) = f'(n_j)$, where $\boldsymbol{B}^{f'} = Pro(\boldsymbol{B}^f \wedge \boldsymbol{K}\Sigma_\Box, \varrho_l(y))$ and $\Sigma_\Box$ is as the construction above. See Appendix A.1 for detailed proofs.

A crucial point in the above reduction is that the actions' likelihood $\mathsf{M}_l(h_s, y)$ might depend on the current state h_s. A natural question is whether the verification problem is decidable if we set the likelihood axiom to be strongly context-free. The following theorem provides a negative answer for this when the SSAs are local-effect.

Theorem 4. *The verification problem is undecidable for programs with local-effect SSAs and strongly context-free likelihood axioms.*

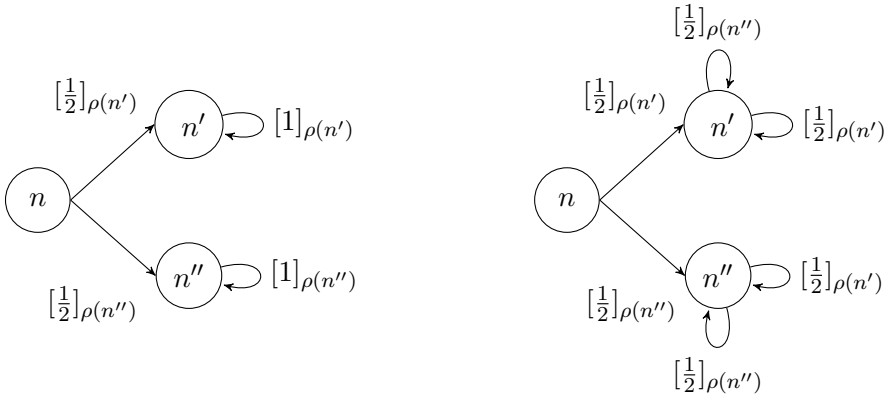

Figure 2: Two possible BATs to simulate an SSPA with a single probabilistic transition.

Since the likelihood axiom is restricted to be strongly context-free, the previous reduction breaks as transition probabilities of probabilistic automata may depend on states. Nevertheless, we reduce the emptiness problem of the *simple probabilistic automata* (SPA) (Gimbert & Oualhadj, 2010), i.e. PA whose transition probabilities are among $\{0, \frac{1}{2}, 1\}$, to the verification problem with context-free likelihood axiom and local-effect SSAs. More precisely, the simple probabilistic automata we consider are *super simple probabilistic automata* (SSPA), that is, SPA with a single probabilistic transition and where every transition has a unique letter. Fijalkow *et al.* (2012a) show that the emptiness problem of the SPA with even a single probabilistic transition is undecidable. In their proofs (Fijalkow, Gimbert, & Oualhadj, 2012b), the single-step transitions of an SPA are simulated by words of another SPA (with a single probabilistic transition). The SPA with a single probabilistic transition is almost super simple except that a letter *merge* is used multiple times when simulating different transitions of the original SPA. We can easily relabel it to make it distinct: by replacing it with new letters $merge(a, q)$ where q records the outgoing state and a records the latter of the transition in the original SPA.

The idea of the reduction is to shift the likelihood context in likelihood axioms to the context formula in SSAs. More concretely, instead of saying the action likelihood depends on the state and the action's effect is fixed, which is the view of the BAT in the previous reduction, we say the action's effect depends on the state and the action's likelihood is fixed. This is better illustrated by an example. Consider an SSPA consisting of a single probabilistic transition with $q \xrightarrow{0.5,l} q'$ and $q \xrightarrow{0.5,l} q''$ as in Fig. 2 where we use standard names n, n', n'' to represent corresponding states q, q', q'' and the primitive program ϱ (with instantiations $\varrho(n')$ and $\varrho(n'')$) to represent the letter l. Clearly, one can construct a BAT as in the previous reduction to simulate this (Fig. 2 left), i.e., ground action $\varrho(n')$ (likewise for $\varrho(n'')$) will deterministically set h_s to n', yet the likelihood of $\varrho(n')$ depends on what the current state is. Nevertheless, a BAT with local-effect SSAs and context-free likelihood axiom can simulate it as well (Fig. 2 right), i.e. the likelihood of $\varrho(n')$ and $\varrho(n'')$ are the same, yet they have different effects conditioned on the current state. In the right part of Fig. 2, both $\varrho(n')$ and $\varrho(n'')$ have the same effect, i.e. stay in state n'. Consequently,

executing ϱ in state n' will end up in n' with likelihood 1 as well. See Appendix A.2 for detailed proofs.

5. A Decidable Case

Another source of undecidability comes from the property specification, more precisely, the unbounded until operators. In fact, in our program semantics, the MDP $\mathsf{M}_\delta^{e,w}$ is indeed an infinite partially observable MDP (POMDP) where the set of observations can be thought of as the set of possible KBs that can be progressed to from the initial KB regarding a certain possible action sequence of the program. Verifying belief programs against specifications with unbounded $\mathbf{U}$ requires verification of infinite-horizon POMDPs, which is known to be undecidable (Norman et al., 2017). This motivates us to focus on the case with only bounded until operators. In contrast to the previous section, we now allow arbitrary domain specifications.

Definition 10. A state formula Φ' is called bounded if it contains no $\mathbf{U}$ and no nested $\mathbf{P}$.

Namely, $\Phi' ::= \beta \mid \mathbf{P}_I[\Psi'] \mid \neg\Phi' \mid \Phi' \wedge \Phi'$ with $\Psi' ::= \mathbf{X}\beta \mid (\beta \mathbf{U}^{\leq k} \beta)$ where β is any static subjective $\mathcal{DS}_p$ formula as in Eq. (9a). For example, the property **P1** $\mathbf{P}_{\geq 0.5}[\mathbf{F}^{\leq 2}\boldsymbol{B}(\mathit{hpos} = 2\colon 1)]$ is bounded while the property **P2** $\mathbf{P}_{=1}[\mathbf{F}\boldsymbol{B}(\mathit{hpos} = 2\colon 1)]$ is not. We leave out nested $\mathbf{P}$ as the model checking algorithm (Norman et al., 2017) on finite POMDPs we rely on do not support it. For bounded state formulas, we only need to consider action sequences with a bounded length, namely, only a finite subset of $\mathsf{M}_\delta^{e,w}$'s states and observations needs to be considered. Although model-checking the finite subset of $\mathsf{M}_\delta^{e,w}$ against PCLT formulas without unbounded $\mathbf{U}$ operators is decidable, this does not entail that the verification problem is decidable as infinitely many such subsets exist. This is because there are infinitely many models (e, w) satisfying the initial state axiom. Our solution is to abstract them into finitely many equivalence classes as in (Zarrieß & Claßen, 2016).

First, we need to identify the so-called *program context* $\mathcal{C}(\mathcal{BP})$ of a given program $\mathcal{BP}$, which contains: (1) all sentences in Σ_0; (2) all likelihood conditions $\phi_j(x)$ (grounded by all possible actions) and ϕ_j in Eq. (6);[9] (3) all test conditions in the program expression; (4) all $\mathcal{DS}_p$ sub-formulas of the temporal formula; (5) the negation of formulas from (1)-(4).

Now, for a given bounded state formula Φ', we define the maximal step $k = \mathit{maxstep}(\Phi')$ for it as (assuming β is a $\mathcal{DS}_p$ subjective formula):

1. $\mathit{maxstep}(\Phi') = 0$ if $\Phi' = \beta$;
2. $\mathit{maxstep}(\Phi') = 1$ if $\Phi' = \mathbf{P}_I[\mathbf{X}\beta]$;
3. $\mathit{maxstep}(\Phi') = k'$ if $\Phi' = \mathbf{P}_I[\beta \mathbf{U}^{\leq k'} \beta]$;
4. $\mathit{maxstep}(\Phi') = \mathit{maxstep}(\Phi'')$ if $\Phi' = \neg\Phi''$;
5. $\mathit{maxstep}(\Phi') = \max\{\mathit{maxstep}(\Phi''), \mathit{maxstep}(\Phi''')\}$ if $\Phi' = \Phi'' \wedge \Phi'''$.

We then define the types of models as follows:

9. Since there are finitely many primitive programs in $\mathcal{BP}$, x has finitely many possible values. Then $\mathcal{C}(\mathcal{BP})$ includes all possible groundings of $\phi_j(x)$.

Definition 11 (Types)**.** Given a belief program $\mathcal{BP}$ and a bounded state formula Φ', let $\mathcal{A}_{\mathcal{BP}}$ be the set of all ground actions with non-zero likelihood in $\mathcal{BP}$, $(\mathcal{A}_{\mathcal{BP}})^k$ be the set of all action sequences using actions in $\mathcal{A}_{\mathcal{BP}}$ with length no greater than k where $k = maxstep(\Phi')$. The set of all type elements is given by:

$$\mathsf{TE}(\mathcal{BP}, \Phi') = \{(z, \alpha) \mid z \in (\mathcal{A}_{\mathcal{BP}})^k, \alpha \in \mathcal{C}(\mathcal{BP})\}$$

A type wrt $\mathcal{BP}$ and Φ' is a set $\tau \subseteq \mathsf{TE}(\mathcal{BP}, \Phi')$ that satisfies:

1. $\forall \alpha \in \mathcal{C}(\mathcal{BP})$, $\forall z \in (\mathcal{A}_{\mathcal{BP}})^k$, $(z, \alpha) \in \tau$ or $(z, \neg\alpha) \in \tau$;
2. there exists e, w s.t. $e, w \models \Sigma_0 \wedge \boldsymbol{B}^f \wedge \boldsymbol{K}\Sigma_\Box \cup \{[z]\alpha \mid (z, \alpha) \in \tau\}$.

Let $\mathsf{Types}(\mathcal{BP}, \Phi')$ denote the set of all types wrt $\mathcal{BP}$ and Φ'. The type of a model (e, w) is given by $\mathsf{type}(e, w) ::= \{(z, \alpha) \in \mathsf{TE}(\mathcal{BP}, \Phi') \mid e, w \models [z]\alpha\}$. $\mathsf{Types}(\mathcal{BP}, \Phi')$ partitions e, w into equivalence classes in the sense that if $\mathsf{type}(e, w) = \mathsf{type}(e', w')$, then $e, w \models [z]\alpha$ iff $e', w' \models [z]\alpha$ for $z \in (\mathcal{A}_{\mathcal{BP}})^k$ and $\alpha \in \mathcal{C}(\mathcal{BP})$.

Thirdly, we use a representation similar to the *characteristic program graph* (Claßen & Lakemeyer, 2008) where nodes are the reachable subprograms $Sub(\delta)$, each of which is associated with a termination condition $\mathsf{Fin}(\delta')$ (the initial node v_0 corresponds to the overall program δ), and where an edge $\delta_1 \xrightarrow{\varrho/\alpha} \delta_2$ represents a transition from δ_1 to δ_2 by the primitive program ϱ if test condition α holds. Moreover, failure conditions are given by $\mathsf{Fail}(\delta') ::= \neg(\mathsf{Fin}(\delta') \vee \bigvee_{\delta' \xrightarrow{\varrho/\alpha} \delta''} \alpha)$.

Lastly, we define a set of atomic propositions $\mathsf{AP} = \{p_\alpha | \alpha \in \mathcal{C}(\mathcal{BP})$ and α is subjective$\}$ one for each subjective $\alpha \in \mathcal{C}(\mathcal{BP})$.

The finite POMDP for a type τ of a program $\mathcal{BP}$ is a tuple $\mathsf{M}^\tau_\delta = \langle \mathsf{S_{fin}}, \mathsf{Act_{fin}}, \mathsf{P_{fin}}, \mathsf{O_{fin}}, \Omega_{\mathsf{fin}}, \mathsf{s^0_{fin}}, \mathsf{L_{fin}} \rangle$ consisting of:

1. $\mathsf{S_{fin}}$: the set of states $\mathsf{S_{fin}} = (\mathcal{A}_{\mathcal{BP}})^k \times Sub(\delta)$;
2. $\mathsf{Act_{fin}}$: the set of primitive programs $\mathsf{Act_{fin}} = \mathsf{Act}$;
3. $\mathsf{P_{fin}}$: the transition function $\mathsf{P_{fin}}(\langle z_1, \delta_1 \rangle, \varrho, \langle z_2, \delta_2 \rangle)$ as
   - $\mathsf{P_{fin}}(\cdot) = r_{i,j}(x)$ (see Eq. (6)) if $|z_1| < k$, $\delta_1 \xrightarrow{\varrho/\alpha} \delta_2$, $(z_1, \alpha) \in \tau$, and for some $a(x, y), r_i(x), \phi_j(x)$, it holds that (likewise for sensing)
     $$\varrho \to a(x, y), z_2 = z_1 \cdot a(x, y), y = r_i(x), (z_1, \phi_j(x)) \in \tau;$$
   - $\mathsf{P_{fin}}(\cdot) = 1$ if $|z_1| = k, \varrho = \mathfrak{f}, z_1 = z_2, \delta_2 = \delta_1$;
   - $\mathsf{P_{fin}}(\cdot) = 1$ if $(z_1, \mathsf{Fin}(\delta_1)) \in \tau, \varrho = \delta_2 = \epsilon$;
   - $\mathsf{P_{fin}}(\cdot) = 1$ if $(z_1, \mathsf{Fail}(\delta_1)) \in \tau, \varrho = \delta_2 = \mathfrak{f}$;
4. $\mathsf{O_{fin}}$: the observations $\mathsf{O_{fin}} = \{Pro(\boldsymbol{B}^f \wedge \boldsymbol{K}\Sigma_\Box, z) | z \in \mathcal{A}^k_{\mathcal{BP}}\}$;

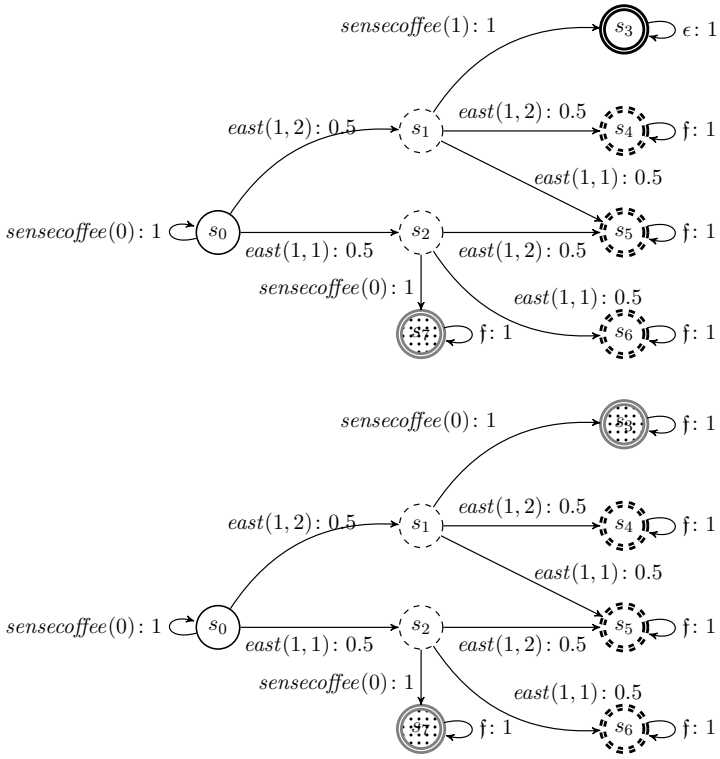

Figure 3: POMDPs induced by type τ_1 (above) and τ_2, τ_3 (below) for the coffee robot example.

5. Ω_{fin} : the state to observation mapping Ω_{fin} as $\Omega_{\mathsf{fin}}(\langle z, \delta\rangle) = Pro(\boldsymbol{B}^f \wedge \boldsymbol{K}\Sigma_\Box, z)$; [10]
6. $\mathsf{s}^0_{\mathsf{fin}}$: the initial state $\mathsf{s}^0_{\mathsf{fin}} = \langle\langle\rangle, \delta\rangle$;
7. $\mathsf{L}_{\mathsf{fin}}$: a labeling function $\mathsf{L}_{\mathsf{fin}}(o) = \{p_\alpha \mid p_\alpha \in \mathsf{AP}, o \models \alpha\}$.[11]

Let AP be a set of atomic propositions, $\mathsf{M} = \langle \mathsf{S}_{\mathsf{fin}}, \mathsf{Act}_{\mathsf{fin}}, \mathsf{P}_{\mathsf{fin}}, \mathsf{O}_{\mathsf{fin}}, \Omega_{\mathsf{fin}}, \mathsf{s}^0_{\mathsf{fin}}, \mathsf{L}_{\mathsf{fin}}\rangle$ a finite POMDP where $\mathsf{L}_{\mathsf{fin}} : \mathsf{O}_{\mathsf{fin}} \mapsto 2^{\mathsf{AP}}$ is a labeling function, Φ_p be a state PCTL formula defined as Eq. (9a) but replacing β with atomic observations in AP, we define the satisfaction relation $\models_p$ among POMDP and propositional PCTL as in (Norman et al., 2017). Namely, for a state $s \in \mathsf{S}_{\mathsf{fin}}$,

10. We focus on deterministic observation functions. More general POMDPs with stochastic observation functions can be transformed into ones with deterministic observation functions in polynomial time (Chatterjee, Chmelik, Gupta, & Kanodia, 2016). Besides, under such a setting, an observation in a POMDP is essentially an equivalence class of states. We call a belief distribution an observation since it represents an equivalence class of action histories in $\mathcal{A}^k_{\mathcal{BP}}$ given by the relation $\approx$ (observational indistinguishability): under our assumption of stochastic actions and sensing, $z \approx z'$ implies that $Pro(\boldsymbol{B}^f \wedge \boldsymbol{K}\Sigma_\Box, z)$ and $Pro(\boldsymbol{B}^f \wedge \boldsymbol{K}\Sigma_\Box, z')$ are equivalent given any initial KB $\boldsymbol{B}^f \wedge \boldsymbol{K}\Sigma_\Box$.
11. Here, we use a function $\mathbf{E}[\mathrm{KB}, \alpha]$ to evaluate a subjective formula against a KB. Essentially, the function is a special case of the regression operator in (Liu & Lakemeyer, 2021) and returns a rigid formula. Thereafter, $\mathrm{KB} \models \alpha$ is reduced to $\models \mathbf{E}[\mathrm{KB}, \alpha]$. For example, let KB be as in Example 1, $\mathbf{E}[\mathrm{KB}, \boldsymbol{B}(h = 2) < 1]$ returns $f_0(2) < 1$. Since $f_0(2) = 0$ and $\models 0 < 1$, $\mathrm{KB} \models \boldsymbol{B}(h = 2) < 1$.

Shape	Observation (Belief distribution $\boldsymbol{B}^f$)
Single circle (s_0)	$\{hpos = 0 \to 1\}$, i.e. $\boldsymbol{B}^{f_0}$
Dashed circle (s_1, s_2)	$\{hpos = 1 \to 0.5, hpos = 2 \to 0.5\}$, i.e. $\boldsymbol{B}^{f_1}$
Dashed double circle (s_4, s_5, s_6)	$\{hpos = 2 \to 0.25, hpos = 3 \to 0.5, hpos = 4 \to 0.25\}$
Double circle with fills (s_7, s_3 below)	$\{hpos = 1 \to 1\}$
Double circles without fills (s_3 above)	$\{hpos = 2 \to 1\}$, i.e. $\boldsymbol{B}^{f_2}$

Table 3: The observations (belief distributions) for states of POMDPs in Fig. 3.

1. $\mathsf{M}, s \models_p p$ iff $p \in \mathsf{L}_{\mathsf{fin}}(\Omega(s))$;
2. $\mathsf{M}, s \models_p \mathbf{P}_I[\Psi]$ iff for all *observation-based* policies σ, $\mathsf{Pr}^\sigma_s(\Psi) \in I$, where

$$\mathsf{Pr}^\sigma_s(\Psi) = \mathsf{Pr}^\sigma_s(\{\pi \in \mathsf{Path}^\sigma(s, \mathsf{M}) \mid \mathsf{M}, \pi \models_p \Psi\}).$$

The rules for other state formulas and trace formulas are the same as in Section 3.2. In addition, a policy σ is called *observation-based* iff for all $s, s' \in \mathsf{S}_{\mathsf{fin}}$, if $\Omega_{\mathsf{fin}}(s) = \Omega_{\mathsf{fin}}(s')$ then $\sigma(s) = \sigma(s')$. With these notions, we have that:

Lemma 1. *Given a program $\mathcal{BP}$ and a bounded state formula Φ', for all e, w s.t. $e, w \models \Sigma \wedge \boldsymbol{B}^f \wedge \boldsymbol{K}\Sigma_\Box$, $\mathsf{M}^{e,w}_\delta \models \Phi'$ iff $\mathsf{M}^\tau_\delta \models_p \Phi'_p$, where τ is the type of e, w, Φ'_p a PCTL formula obtained from Φ' by replacing all its $\mathcal{DS}_p$ sub-formula with the corresponding atomic proposition (Proofs in Appendix A.3).*

The lemma suggests that the verification problem can be reduced to model-check a finite POMDP against propositional PCTL properties. Since there are only finitely many type elements, there are only finitely many types for a given program. Hence, we can exploit existing model-checking tools like Prism (Kwiatkowska et al., 2011) or Storm (Hensel et al., 2022) to verify the PCTL properties against these finitely many POMDPs. Consequently, we have the following theorem.

Theorem 5. *The verification problem is decidable for temporal properties specified by bounded state formulas.*

In our coffee robot example, for the belief program $\mathcal{BP}_{cfe}$ in Eq. (8) and property **P1** $\mathbf{P}_{\geq 0.5}[\mathbf{F}^{\leq 2}\boldsymbol{B}(hpos = 2 \colon 1)]$, we obtain three types τ_1, τ_2, and τ_3 for worlds satisfying $\{hpos = 0\}$, $\{hpos = -1\}$, and $\{hpos < 0 \wedge hpos \neq -1\}$ in the initial state $\mathsf{O}_{\mathsf{fin}}$, respectively. This is because Σ_0 only says $\{hpos \leq 0\}$. The corresponding finite POMDPs are depicted in Fig. 3. Note that the POMDPs for τ_2, τ_3 are the same. The observations of states are indicated by shape and the corresponding observations (belief distributions of *hpos*) are given in Table 5. For the property **P1** $\mathbf{P}_{\geq 0.5}[\mathbf{F}^{\leq 2}\boldsymbol{B}(hpos = 2 \colon 1)]$, we only need to consider the first two choices of proper policies, hence there are four equivalence classes of proper policies: $\{east(1) \cdot east(1), east(1) \cdot sensecoffee, sensecoffee \cdot east(1), east(1) \cdot sensecoffee\}$, it is not hard to check that only the proper policies in the last class in the POMDP of τ_1, i.e. starting with $east(1) \cdot sensecoffee$ can reach the observation $\boldsymbol{B}^{f_2} \wedge \boldsymbol{K}\Sigma_\Box$ which satisfies the label $p_{\boldsymbol{B}(hpos=2 \colon 1)}$ with a probability $0.5 \times 1 = 0.5$. Therefore, $\mathsf{M}^{\tau_1}_{\delta_{cfe}} \not\models_p \mathbf{P}_{\geq 0.5}[\mathbf{F}^{\leq 2} p_{\boldsymbol{B}(hpos=2 \colon 1)}]$ as the semantics of $\models_p$ considers all proper policies. Hence, $\mathcal{BP} \not\models \mathbf{P}_{\geq 0.5}[\mathbf{F}^{\leq 2}\boldsymbol{B}(hpos = 2 \colon 1)]$ as the semantics of $\models$ in Def. 7 requires all the underlying POMDPs to satisfy the property.

6. Related Work

Our formalism extends the modal logic $\mathcal{DS}_p$ (Liu & Feng, 2023), a variant of the logic $\mathcal{DS}$ (Belle & Lakemeyer, 2017). The idea of using the same modal logic to specify the program and its properties is inspired by the work of Claßen and Zarrieß (2017). Similar approaches to the verification of CTL*, LTL, and CTL properties of Golog programs include the works (Claßen & Lakemeyer, 2008), (Zarrieß & Claßen, 2015), and (Zarrieß & Claßen, 2016) respectively. Axiomatic approaches to the verification of Golog programs can be found in works (Giacomo et al., 1997, 2016, 2020). While the difference in formalisms of programs do not affect the computational complexity of verification in principle, it does impact the approach of verification, for example, axiomatic approaches tend to use theorem-proving (Lin, 2016; Giacomo et al., 2016) to perform verification, while the above modal approaches tend to use model-checking or similar techniques for verification. Another comment is that the difference in program properties that need to be verified indeed influences the computational complexity of verification. This is evidenced by propositional model-checking for finite transition systems as in (Baier & Katoen, 2008), where it is shown that the model-checking problem for LTL, CTL, and CTL* is PSPACE-complete, PTIME, and PSPACE-complete, respectively.

While the verification of arbitrary Golog programs is clearly undecidable due to the underlying first-order logic, Claßen *et al.* (2014) established decidability in case the underlying logic is restricted to the two-variable fragment, the program constructs disallow non-deterministic pick of action arguments, and the BATs are restricted to be local-effect. Later, the constraints on BATs are relaxed to acyclic and flat BATs (Zarrieß & Claßen, 2016). Under similar settings, the works (Zarrieß & Claßen, 2015; Claßen & Zarrieß, 2017) show that the verification of $\mathcal{ALCOK}$-Golog programs, where the underlying logic is a description logic, and DT-Golog programs against LTL and PRCTL specification, respectively, is decidable. What distinguishes our work from the above is that we assume that the environment is partially observable by the agent, while they assume full observability.

Verifying temporal properties under partial observation has been studied extensively in model checking (Chatterjee et al., 2016; Norman et al., 2017; Bork et al., 2020), in planning (Madani et al., 2003), and in stochastic games (Kwiatkowska et al., 2009). Notably the work on probabilistic planning (Madani et al., 2003) is closely related to our belief program verification as belief programs can be viewed as a compact representation of a plan. Moreover, the paper suggested that probabilistic planning is undecidable under different restrictions. Perhaps the most relevant restriction is that probabilistic planning is undecidable even without observations, which essentially corresponds to our restriction on context-free likelihood axioms that exclude sensing. However, our results go beyond this as we show the problem remains undecidable when restricting actions to be local-effect. Another proposal on compact representations of plans is the belief programs by Lang and Zanuttini (2015). Nevertheless, the proposal is limited as the underlying logic is propositional. Hence, verification there reduces to regular model-checking. In contrast, our framework is based on the logic $\mathcal{DS}_p$ which allows us to express incompleteness about the underlying model. Therefore, to verify a belief program, one has to perform model-checking for potentially infinitely many POMDPs. Another advantage of our belief programs is that tests of the program can refer to beliefs about beliefs (meta-beliefs) and beliefs with quantifying-in. Hence, although

Lang and Zanuttini (2015) showed that the verification problem is decidable when restricted to a finite horizon, our decidability result is more general than theirs.

Finally, we remark that verification in robotics has been an active area of research for a long time, see (Luckcuck et al., 2019) for a recent survey. Approaches that deal with stochastic actions predominantly employ probabilistic model checkers such as PRISM (Kwiatkowska et al., 2011) or Storm (Hensel et al., 2022). An example similar in spirit to our approach is (Izzo et al., 2016). Here robot programs written in a variant of the agent programming language *Jason* (Bordini et al., 2007) are translated into Discrete-time Markov Chains and MDPs. Temporal properties formulated in PCTL are then verified using PRISM. In contrast to our framework, the computational complexity is not analyzed and POMDPs are not considered.

7. Conclusion

We reconsider the proposal of belief programs by Belle and Levesque based on the logic $\mathcal{DS}_p$. The main contribution of the paper is the study of the complexity of the verification problem. As it turns out, the problem is undecidable even in very restrictive settings. However, we also show a case where the problem is indeed decidable.

As for future work, there are two promising directions. Regarding the complexity of verification, whether it is decidable or not remains open for the case where the SSAs and likelihood axiom are strongly context-free. We conjecture that, under such a setting, belief programs, in general, cannot simulate arbitrary probabilistic automata, but only a subset. Since the emptiness problem of probabilistic automata can be viewed as a special case of the verification problem, showing the undecidability of the emptiness problem for such a subset would prove the undecidability of the verification problem for programs with strongly context-free SSAs and likelihood axioms. Besides, The works (Chatterjee & Tracol, 2012; Fijalkow et al., 2012a) show a set of decidable decision problems related to special types of probabilistic automata. It is interesting to see how these problems can be transformed into verification problems and hence find decidable cases. Another direction is more practical. It is desirable to design a general algorithm to perform verification of arbitrary belief programs, even if the algorithm might not terminate. In this regard, based on our new semantics of belief programs and techniques for solving first-order MDP and first-order POMDP (Sanner & Boutilier, 2009; Sanner & Kersting, 2010), Liu *et al.* (2023b) proposed a symbolic dynamic programming algorithm to verify *reachability probability*. It would be desirable to extend it for arbitrary LTL or PCTL-like properties.

Acknowledgments

Daxin was also affiliated with RWTH Aachen University; This work has been supported by the Deutsche Forschungsgemeinschaft (DFG, German Research Foundation) RTG 2236/2 'UnRAVeL' and by the EU ICT-48 2020 project TAILOR (No. 952215).

Appendix A. Proofs

A.1 Proof of Theorem 3

Theorem 3 The verification problem is undecidable for programs with strongly context-free SSAs.

Before turning to the proofs, we introduce some notations. A *probability distribution* on a set of (finite) states Q is a mapping $o : \mathsf{Q} \mapsto [0, 1]$ such that $\sum_{q\in\mathsf{Q}} o(q) = 1$. The set $\{q \in \mathsf{Q} | o(q) > 0\}$ is called the support of o and denoted as $\mathsf{Supp}(o)$. We write $\mathsf{D}(\mathsf{Q})$ for the set of all probability distributions on Q and use o_q to denote the distribution $o(q) = 1$. We show the undecidability by a reduction of the undecidable emptiness problem for probabilistic automata (Paz, 2014).

Given a PA $\mathcal{A} = (\mathsf{Q}, \mathsf{L}, (\mathsf{M}_l)_{l\in\mathsf{L}}, q_1, \mathsf{F})$, we construct a belief program to simulate its run. More formally, we use a single fluent h_s to record the current state, a set of standard names $\mathcal{N}_\mathsf{Q} = \{n_1, n_2...n_{|\mathsf{Q}|}\}$ to represent the states in Q, a set of stochastic actions $\varrho_i(y)$ to simulate the read of letter $l_i \in \mathsf{L}$, for BAT Σ, we have $\Sigma := \Sigma_0 \cup \Sigma_\Box$ with $\Sigma_0 = \{h_s = n_1\}$ and $\Sigma_\Box$ as

$$\begin{aligned} \Box[a]h_s = u \equiv & \bigvee_i a = \varrho_i(u) \\ & \vee h_s = u \wedge \bigwedge_i \forall u'.a \neq \varrho_i(u') \\ \Box l(\varrho_i(y)) = u \equiv & \bigvee_{j,j'\in\{1,\ldots,|\mathsf{Q}|\}} y = n_{j'} \wedge h_s = n_j \wedge u = \mathsf{M}_{l_i}[j, j'] \end{aligned} \tag{10}$$

Here, $\mathsf{M}_{l_i}[j, j']$ refers to the scalar value of entry $[j, j']$ of the matrix M_{l_i}. One can check that the likelihood axioms above indeed satisfy the assumptions we imposed as in Eq. (6).

Clearly, the SSA is strongly context-free. Intuitively, the BAT says that fluent h_s can only be changed by action $\varrho_i(y)$ and the unobservable argument y determines the new state; the likelihood of $\varrho_i(y)$ depends on the current state h_s and equals the transition probability $\mathsf{M}_{l_i}(h_s, y)$.

Lemma 2. *Given a PA $\mathcal{A}$ and BAT Σ as above, $o \in \mathsf{D}(\mathsf{Q})$. Let KB be as $\boldsymbol{B}^f \wedge \boldsymbol{K}\Sigma_\Box$ s.t. $f(n_k) = o(q_k)$ for $1 \leq k \leq |\mathsf{Q}|$, then for all k s.t. $1 \leq k \leq |\mathsf{Q}|$ and all i:*

$$f'(n_k) = (o \cdot l_i)(q_k) \text{ if } \boldsymbol{B}^f \wedge \boldsymbol{K}\Sigma_\Box \models [\varrho_i(n)]\boldsymbol{B}^{f'} \wedge \boldsymbol{K}\Sigma_\Box \text{ for some } n \in \mathcal{N}_\mathsf{O}.$$

Namely, if the initial belief distribution $\boldsymbol{B}^f$ coincides with the initial distribution over states o of the probabilistic automaton $\mathcal{A}$, then the progressed belief distribution $\boldsymbol{B}^{f'}$ of $\boldsymbol{B}^f$ in terms of $\varrho_i(n)$ coincides with the derived distribution over states $(o \cdot l_i)$.

Proof. The proof is straightforward. According to Eq.(4) in the progression theorem,

$$\begin{aligned} f'(u) &= \sum_{u' \in \mathbb{D}} f(u') \times \sum_{y' \in \mathbb{D}} l(\varrho_i(y')) \times \mathbb{I}(u, u', \varrho_i(y')) \\ &= \sum_{u' \in \mathcal{N}_{\mathsf{Q}}} f(u') \sum_{y' \in \mathcal{N}_{\mathsf{Q}}} \mathsf{M}_{l_i}(u', y') \times \begin{cases} 1 & u = y' \\ 0 & otherwise \end{cases} \\ &= \sum_{u' \in \mathcal{N}_{\mathsf{Q}}} f(u') \mathsf{M}_{l_i}(u', u) \end{aligned}$$

The second equality is because $f(u')$ is non-zero only if $u' \in \mathcal{N}_{\mathsf{Q}}$ and the definition of $\mathcal{L}(a)$. In addition, since $Pro(h_s = u', \varrho_i(y')) = \{h_s = y'\}$, the indicator function $\mathbb{I}$ reduce to **if** $u = y'$ **then** 1 **else** 0. Since $f(n_k) = o(q_k)$ for all $1 \leq k \leq |\mathsf{Q}|$ by assumption,
$f'(n_k) = \sum_{u' \in \mathcal{N}_{\mathsf{Q}}} f(u') \mathsf{M}_{l_i}(u', n_k) = \sum_{q_k \in \mathsf{Q}} o(q_k) \times \mathsf{M}_{l_i}(q_k, n_k) = (o \cdot l_i)(q_k)$. □

The lemma can be easily extended from actions to action sequences. As a result, we have:

Proposition 1. *Given a PA $\mathcal{A}$ and BATs Σ as above, $o \in \mathsf{D}(\mathsf{Q})$. Let KB be as $\boldsymbol{B}^f \wedge \boldsymbol{K}\Sigma_{\Box}$ s.t. $f(n_i) = o(q_i)$ for $1 \leq i \leq |\mathsf{Q}|$, additionally, let z be any action sequence of ground actions with action types in $\{\varrho_1, \varrho_2, \ldots, \varrho_{|\mathsf{L}|}\}$, $\mathbf{w}$ be the corresponding word of z, then for all k s.t. $1 \leq k \leq |\mathsf{Q}|$,*

$$f'(n_k) = (o \cdot \mathbf{w})(q_k) \text{ if } \boldsymbol{B}^f \wedge \boldsymbol{K}\Sigma_{\Box} \models [z]\boldsymbol{B}^{f'} \wedge \boldsymbol{K}\Sigma_{\Box}.$$

Now let $\boldsymbol{B}^f$ be as $\boldsymbol{B}(h_s = n_1 \colon 1)$, then the program $\mathcal{BP} = (\Sigma \cup \boldsymbol{B}^f \wedge \boldsymbol{K}\Sigma_{\Box}, \delta)$ simulates the run of PA $\mathcal{A}$ where

$$\delta := \textbf{while } \boldsymbol{B}(h_s \in \mathcal{N}_{\mathsf{F}}) < \xi \textbf{ do } \varrho_1 \,|\varrho_2, \ldots, |\, \varrho_{|\mathsf{L}|} \textbf{ endWhile}.$$

Here $\mathcal{N}_{\mathsf{F}}$ is the set of standard names representing the accepting states F in $\mathcal{A}$, ϱ_i is the primitive program corresponding to action $\varrho_i(y)$.

Clearly, given e, w s.t. $e, w \models \Sigma \wedge \boldsymbol{B}^f \wedge \boldsymbol{K}\Sigma_{\Box}$, a proper policy σ of $\mathsf{M}_{\delta}^{e,w}$ can be represented by a (possibly infinite) sequence $\vec{\varrho}$ of primitive programs in $\{\varrho_1, \ldots, \varrho_{|\mathsf{L}|}\}$.[12]

Lemma 3. *Given a PA $\mathcal{A}$, a belief program $\mathcal{BP}$ as above, and e, w s.t. $e, w \models \Sigma \wedge \boldsymbol{B}^f \wedge \boldsymbol{K}\Sigma_{\Box}$, let $\sigma = \vec{\varrho}$ be a proper policy of $\mathsf{M}_{\delta}^{e,w}$. Then for any $\xi \in [0, 1]$,*

$$\mathsf{Pr}_{s_0}^{\sigma}(\mathbf{G}\boldsymbol{B}(h_s \in \mathcal{N}_{\mathsf{F}}) < \xi) = 1 \text{ iff } \forall \mathbf{w} \in Fin(\vec{\varrho}), \mathbb{P}_{\mathcal{A}}(q_1 \xrightarrow{\mathbf{w}} \mathsf{F}) < \xi,$$

where $Fin(\vec{\varrho})$ is the set of all finite prefixes of the infinite world that corresponds to $\vec{\varrho}$ in $\mathcal{A}$.

Proof. $\mathsf{Pr}_{s_0}^{\sigma}(\mathbf{G}\boldsymbol{B}(h_s \in \mathcal{N}_{\mathsf{F}}) < \xi) = 1$ iff (by Def. of $\mathsf{Pr}_{s_0}^{\sigma}$)
for all $\pi \in \sigma\text{-}paths$, $\mathsf{M}_{\delta}^{e,w}, \pi \models \mathbf{G}\boldsymbol{B}(h_s \in \mathcal{N}_{\mathsf{F}}) < \xi$) iff (by semantics of $\mathbf{G}$)
for all reachable configurations s under σ, $\mathsf{M}_{\delta}^{e,w}, s \models \boldsymbol{B}(h_s \in \mathcal{N}_{\mathsf{F}}) < \xi$ iff (since every reachable configurations $s = \langle z, \delta \rangle$, z is an instantiation of $\vec{\varrho}$'s finite prefix and vise verse)
for all z s.t z is an instantiation of $\vec{\varrho}$'s finite prefix, $e, w, z \models \boldsymbol{B}(h_s \in \mathcal{N}_{\mathsf{F}}) < \xi$ iff (by Prop. 1) $\forall \mathbf{w} \in Fin(\vec{\varrho}), \mathbb{P}_{\mathcal{A}}(\mathbf{w}) < \xi$. □

12. If the program terminates then $\vec{\varrho}$ is finite.

As an easy consequence:

Proposition 2. *Given a PA $\mathcal{A}$, a threshold $\xi \in [0,1]$, and a belief program $\mathcal{BP}$ as above, $\mathcal{BP} \models \mathbf{P}_{=1}[\mathbf{GB}(h_s \in \mathcal{N}_{\mathsf{F}}) < \xi]$ iff $\mathbb{P}_{\mathcal{A}}(\mathbf{w}) < \xi$ for all word $\mathbf{w}$ of $\mathcal{A}$.*

Proof. **of Theorem 3**

Since, the emptiness problem is undecidable, whether $\mathbb{P}_{\mathcal{A}}(\mathbf{w}) < \xi$ for all word $\mathbf{w}$ of a given PA $\mathcal{A}$ and threshold ξ is undecidable. Therefore, $\mathcal{BP} \models \mathbf{P}_{=1}[\mathbf{GB}(h_s \in \mathcal{N}_{\mathsf{F}}) < \xi]$ is undecidable where $\mathcal{BP}$ is constructed as above by Proposition 2. □

A.2 Proof of Theorem 4

Theorem 4 The verification problem is undecidable for programs with local-effect SSAs and strongly context-free likelihood axioms.

Since the likelihood axioms are restricted to be strongly context-free, the previous reduction does not work anymore. Nevertheless, one can reduce the emptiness problem of the *simple probabilistic automaton* (SPA), i.e. PAs whose transition probability is among $\{0, \frac{1}{2}, 1\}$, to the verification problem with context-free likelihood axioms and local-effect SSAs. More precisely, the simple probabilistic automata we consider are super simple probabilistic automata (SSPA), that is, SPA with a single probabilistic transition and every transition has a unique letter. Fijalkow *et al.* (2012a) show that, given $\xi \in [0,1]$, for any PA $\mathcal{A}$ there is an SPA $\mathcal{A}'$ such that, if there exists a word $\mathbf{w}$ in $\mathcal{A}$ with $\mathbb{P}_{\mathcal{A}}(q_1 \xrightarrow{\mathbf{w}} \mathsf{F}) \geq \xi$, then there exists a word $\mathbf{w}'$ in $\mathcal{A}'$ with $\mathbb{P}_{\mathcal{A}'}(q_1 \xrightarrow{\mathbf{w}'} \mathsf{F}) \geq \xi$. Their construction of the SPA contains exactly one probabilistic transition served as the random bit. By renaming letters in their SPA, one can obtain an SSPA.

The idea of the reduction is to shift the likelihood context in likelihood axioms to the context formula in SSAs. More concretely, instead of saying an action's likelihood depends on the state and the action's effect is fixed, which is the view of BATs in the previous reduction, we say the action's effect depends on the state and the action's likelihood is fixed.

More formally, a SSPA is a six-tuple $\mathcal{A} = (\mathsf{Q}, \mathsf{L}, \mathsf{T}_{\mathsf{P}}, \mathsf{T}_{\mathsf{D}}, q_1, \mathsf{F})$, where $\mathsf{Q}, \mathsf{L}, q_1, \mathsf{F}$ are as before. T_{P} are two pairs $\langle q^\star, q' \rangle$ and $\langle q^\star, q'' \rangle$ $(q', q'', q^\star \in \mathsf{Q})$ representing the unique probabilistic transition. $\mathsf{T}_{\mathsf{D}} \subseteq \mathsf{Q} \times \mathsf{L} \times \mathsf{Q}$ is a finite set of triples of the form $\langle q_i, l_{i,j}, q_j \rangle$ specifying the remaining deterministic transitions.

Now given an SSPA $\mathcal{A}$, we design the following program to simulate it. First, our BAT includes a single fluent h_s and the finite set of standard names $\mathcal{N}_{\mathsf{Q}}$ as before. We use a stochastic action $\varrho_\star(y)$ to simulate the unique probabilistic letter $\star$. For the remaining deterministic transition $\langle q_i, l_{i,j}, q_j \rangle$, they are simulated by stochastic action $\varrho_{i,j}(y)$. The

BAT Σ is given by $\Sigma := \Sigma_0 \cup \Sigma_\Box$ with $\Sigma_0 = \{h_s = n_1\}$ and $\Sigma_\Box$ as

$$\begin{aligned}\Box[a]h_s = u \equiv & a = \varrho_\star(u) \wedge h_s = n^\star \\ & \vee \bigvee_{i,j} a = \varrho_{i,j}(u) \wedge h_s = n_i \\ & \vee h_s = u \wedge (\forall y.a \neq \varrho_\star(y) \vee h_s \neq n^\star) \wedge \bigwedge_{i,j} (\forall y.a \neq \varrho_{i,j}(y) \vee h_s \neq n_i) \qquad (11)\end{aligned}$$

$$\Box l(\varrho_\star(y)) = u \equiv y = n' \wedge u = \frac{1}{2} \vee y = n'' \wedge u = \frac{1}{2}$$

$$\Box l(\varrho_{i,j}(y)) = u \equiv y = n_j \wedge u = 1$$

Clearly, the SSA is local-effect. Intuitively, the BAT says that fluent h_s can be changed by action $\varrho_\star(y)$ or $\varrho_{i,j}(y)$ and the unobservable argument y determines the new state; the likelihood of action is fixed: $\varrho_\star(y)$ always results in $\varrho_\star(n'), \varrho_\star(n'')$ with half-half likelihood and $\varrho_{i,j}(y)$ always results in $\varrho_{i,j}(n_j)$, where n_j is the corresponding standard name of q_j (the successor of q_i); the effects of $\varrho_\star(y)$ and $\varrho_{i,j}(y)$ depend on the current state h_s: $\varrho_\star(y)$ only has effects if the current state is exactly the probabilistic transition state, likewise $\varrho_{i,j}(y)$ have effects if the current state is at the source state q_i.

With the above construction, we have a lemma similar to Lemma 2.

Lemma 4. *Given a SSPA $\mathcal{A}$ and BATs Σ as above, $o \in \mathsf{D}(\mathsf{Q})$. Let KB be as $\boldsymbol{B}^f \wedge \boldsymbol{K}\Sigma_\Box$ s.t. $f(n_k) = o(q_k)$ for $1 \leq k \leq |\mathsf{Q}|$, then for all k s.t. $1 \leq k \leq |\mathsf{Q}|$,*

1. *$f'(n_k) = (o \cdot l_\star)(q_k)$ if $\boldsymbol{B}^f \wedge \boldsymbol{K}\Sigma_\Box \models [\varrho_\star(n)]\boldsymbol{B}^{f'} \wedge \boldsymbol{K}\Sigma_\Box$ for some $n \in \mathcal{N}_\mathsf{Q}$;*
2. *$f'(n_k) = (o \cdot l_{i,j})(q_k)$ if $\boldsymbol{B}^f \wedge \boldsymbol{K}\Sigma_\Box \models [\varrho_{i,j}(n)]\boldsymbol{B}^{f'} \wedge \boldsymbol{K}\Sigma_\Box$ for some $n \in \mathcal{N}_\mathsf{Q}$.*

Proof. Again, the proof is by the progression theorem. Here we only prove the first item. The second can be proved likewise. According to Eq.(4) in the progression theorem,

$$\begin{aligned}f'(u) &= \sum_{u' \in \mathbb{D}} f(u') \times \sum_{y' \in \mathbb{D}} l(\varrho_\star(y')) \times \mathbb{I}(u, u', \varrho_i(y')) \\ &= \sum_{u' \in \mathcal{N}_\mathsf{Q}} f(u') \sum_{y'} \begin{cases} \frac{1}{2} & y' \in \{n', n''\} \\ 0 & \textit{otherwise} \end{cases} \times \begin{cases} 1 & u = y' \wedge u' = n^\star \vee u = u' \wedge u' \neq n^\star \\ 0 & \textit{otherwise} \end{cases}\end{aligned}$$

On the other hand, as for $(o \cdot l_\star)(q_k)$, there are four cases $q_k = q'$, $q_k = q''$, $q_k = q^*$, and $q_k \notin \{q^*, q', q''\}$. The first two cases are similar.

1. **Case $q_k = q'$:** $(o \cdot l_\star)(q_k) = \frac{1}{2}o(q^\star) + o(q')$, the probability of being in q' after reading $l_\star$ amounts to the probability of being in q' beforehand plus a half probability of being in $q^\star$;
2. **Case $q_k = q^\star$:** $(o \cdot l_\star)(q_k) = 0$. The probability mass of $q^\star$ will shift to q' and q'' with half-half likelihood, therefore, after reading $l_\star$, $(o \cdot l_\star)(q_k) = 0$.
3. **Case $q_k \notin \{q^*, q', q''\}$:** $(o \cdot l_\star)(q_k) = o(q_k)$. Nothing changed for such states after reading $l_\star$.

Correspondingly, for $f'(u)$, there are four cases as well, we only show one case of them, the other cases are similar:

1. **Case** $u = n'$**:**

$$\begin{aligned} f'(n') &= \sum_{u' \in \mathcal{N}_{\mathsf{Q}}} f(u') \sum_{y'} \left\{ \begin{array}{ll} \frac{1}{2} & y' \in \{n', n''\} \\ 0 & o/w \end{array} \right. \times \left\{ \begin{array}{ll} 1 & n' = y' \wedge u' = n^\star \\ & \vee n' = u' \wedge u' \neq n^\star \\ 0 & otherwise \end{array} \right. \\ &= f(n^\star) \times \frac{1}{2} + f(n') \times \frac{1}{2} + f(n') \times \frac{1}{2} \\ &= \frac{1}{2} o(q^\star) + o(q') = (o \cdot l_\star)(q'). \end{aligned}$$

   The second equality is because the last indicator is non-zero only if $u' = n^\star$ or $u' = n'$. The third is by assumption $f(q_k) = o(q_k)$ for all k.

This completes the proof. □

As an easy consequence, we have a similar proposition to Prop. 1.

Proposition 3. *Given a SSPA $\mathcal{A}$ and BAT Σ as above, $o \in \mathsf{D}(\mathsf{Q})$. Let KB be as $\boldsymbol{B}^f \wedge \boldsymbol{K}\Sigma_\Box$ s.t. $f(n_i) = o(q_i)$ for $1 \leq i \leq |\mathsf{Q}|$, additionally, let z be any action sequence of ground actions with action types in $\{\varrho_\star, \ldots, \varrho_{i,j}, \ldots\}$, $\mathbf{w}$ be the corresponding word of z, then for all k s.t. $1 \leq k \leq |\mathsf{Q}|$,*

$$f'(n_k) = (o \cdot \mathbf{w})(q_k) \text{ if } \boldsymbol{B}^f \wedge \boldsymbol{K}\Sigma_\Box \models [z]\boldsymbol{B}^{f'} \wedge \boldsymbol{K}\Sigma_\Box.$$

Now let $\boldsymbol{B}^f$ be as $\boldsymbol{B}(h_s = n_1 \colon 1)$, then the program $\mathcal{BP} = (\Sigma \cup \boldsymbol{B}^f \wedge \boldsymbol{K}\Sigma_\Box, \delta)$ simulates the run of PA $\mathcal{A}$ where

$$\delta ::= \textbf{while } \boldsymbol{B}(h_s \in \mathcal{N}_{\mathsf{F}}) < \xi \textbf{ do } \varrho_\star | \ldots | \varrho_{i,j} | \ldots \textbf{ endWhile}.$$

Here $\mathcal{N}_{\mathsf{F}}$ is the set of standard names representing the accepting states F in $\mathcal{A}$, ϱ_i are the primitive program of action $\varrho_i(y)$.

Proposition 4. *Given a SSPA $\mathcal{A}$, a threshold $\xi \in [0, 1]$, and a belief program $\mathcal{BP}$ as above, $\mathcal{BP} \models \mathbf{P}_{=1}[\mathbf{G}\boldsymbol{B}(h_s \in \mathcal{N}_{\mathsf{F}}) < \xi]$ iff $\mathbb{P}_{\mathcal{A}}(\mathbf{w}) < \xi$ for all word $\mathbf{w}$ of $\mathcal{A}$.*

The proof is rather simiar to its counter-part in Prop. 2.

Proof. **of Theorem 4**

Since, the emptiness problem is undecidable for SSPA, whether $\mathbb{P}_{\mathcal{A}}(\mathbf{w}) < \xi$ for all word $\mathbf{w}$ of a given SSPA $\mathcal{A}$ and threshold ξ is undecidable. Therefore, $\mathcal{BP} \models \mathbf{P}_{=1}[\mathbf{G}\boldsymbol{B}(h_s \in \mathcal{N}_{\mathsf{F}}) < \xi]$ is undecidable where $\mathcal{BP}$ is constructed as above by Proposition 4. □

A.3 Proof of Lemma 1

Lemma 1 Given a program $\mathcal{BP}$ and a bounded state formula Φ', for all e, w s.t. $e, w \models \Sigma \wedge \boldsymbol{B}^f \wedge \boldsymbol{K}\Sigma_\Box$, $\mathsf{M}_\delta^{e,w} \models \Phi'$ iff $\mathsf{M}_\delta^\tau \models_p \Phi'_p$, where τ is the type of e, w, Φ'_p a PCTL formula obtained from Φ' by replacing all its $\mathcal{DS}_p$ sub-formula with the corresponding atomic proposition.

Before turning to the proof, we show that our notion of $\mathsf{M}_\delta^{e,w}$ is indeed a POMDP. Consider the set of observations $\mathsf{O}' = \{Pro(\boldsymbol{B}^f \wedge \boldsymbol{K}\Sigma_\Box, z) | z \in (\mathcal{N}_A)^*\}$, and the state to observation mapping Ω as $\Omega(\langle z, \delta\rangle) = Pro(\boldsymbol{B}^f \wedge \boldsymbol{K}\Sigma_\Box), z)$, clearly, $\mathsf{POM} = (\mathsf{S}, \mathsf{P}, \mathsf{Act}, \mathsf{O}, \Omega, s_0)$ forms a POMDP where $\mathsf{S}, \mathsf{P}, \mathsf{Act}, s_0$ is the same as in $\mathsf{M}_\delta^{e,w}$, and the set of proper policies in $\mathsf{M}_\delta^{e,w}$ is exactly the set of observation-based policy (Norman et al., 2017). If the context is clear, we still use $\mathsf{M}_\delta^{e,w}$ to refer to the POMDP POM.

The proof is based on two facts: 1. for a given bounded state formula Φ', we only need to examine finitely many states and observations $\mathsf{M}_{\mathsf{Fin},\delta}^{e,w}$ of $\mathsf{M}_\delta^{e,w}$. 2) The propositional POMDP $\mathsf{M}_\delta^\tau \models \Phi'$ is an abstraction of $\mathsf{M}_{\mathsf{Fin},\delta}^{e,w}$, i.e. $\mathsf{M}_{\mathsf{Fin},\delta}^{e,w} \models \Phi'$ iff $\mathsf{M}_\delta^\tau \models_p \Phi'_p$ by the notion of probabilistic bisimulation.

Recall that for a given bounded state formula Φ', the maximal step $k = maxstep(\Phi')$ for ϕ' is defined as:

1. $maxstep(\Phi') = 0$ if $\Phi' = \beta$;
2. $maxstep(\Phi') = 1$ if $\Phi' = \mathbf{P}_I[\mathbf{X}\beta]$;
3. $maxstep(\Phi') = k'$ if $\Phi' = \mathbf{P}_I[\beta \mathbf{U}^{\leq k'} \beta]$;
4. $maxstep(\Phi') = maxstep(\Phi'')$ if $\Phi' = \neg\Phi''$;
5. $maxstep(\Phi') = \max\{maxstep(\Phi''), maxstep(\Phi''')\}$ if $\Phi' = \Phi'' \wedge \Phi'''$.

Basically, $\mathsf{M}_{\mathsf{Fin},\delta}^{e,w}$ is just the POMDP obtained form $\mathsf{M}_\delta^{e,w}$ by cutting off all states after k steps and making states in the last step absorbing. Trivially, we have $\mathsf{M}_\delta^{e,w} \models \Phi'$ iff $\mathsf{M}_{\mathsf{Fin},\delta}^{e,w} \models \Phi'$.

Now consider the corresponding finite POMDP for a type $\tau = \mathsf{type}(e, w)$, $\mathsf{M}_\delta^\tau = \langle \mathsf{S}_{\mathsf{fin}}, \mathsf{s}_{\mathsf{fin}}^0, \mathsf{A}_{\mathsf{fin}}, \mathsf{P}_{\mathsf{fin}}, \mathsf{O}_{\mathsf{fin}}, \Omega_{\mathsf{fin}}, \mathsf{L}_{\mathsf{fin}}\rangle$ consisting of:

1. the set of states $\mathsf{S}_{\mathsf{fin}} = (\mathcal{A}_{\mathcal{BP}})^k \times Sub(\delta)$;
2. the initial state $\mathsf{s}_{\mathsf{fin}}^0 = \langle\langle\rangle, \delta\rangle$;
3. the set of primitive programs $\mathsf{A}_{\mathsf{fin}} = \mathsf{Act}$;
4. the transition function $\mathsf{P}_{\mathsf{fin}}(\langle z_1, \delta_1\rangle, \varrho, \langle z_2, \delta_2\rangle)$ as

   - $\mathsf{P}_{\mathsf{fin}}(\cdot) = r_{i,j}(x)$ if $|z_1| < k$, $\delta_1 \xrightarrow{\varrho/\alpha} \delta_2$, $(z_1, \alpha) \in \tau$, and for some $a(x, y), r_i(x), \phi_j(x)$, it holds that (likewise for sensing)

     $$\varrho \rightarrow a(x, y), z_2 = z_1 \cdot a(x, y), y = r_i(x), (z_1, \phi_j(x)) \in \tau;$$

   - $\mathsf{P}_{\mathsf{fin}}(\cdot) = 1$ if $|z_1| = k, \varrho = \mathfrak{f}, z_1 = z_2, \delta_2 = \delta_1$;
   - $\mathsf{P}_{\mathsf{fin}}(\cdot) = 1$ if $(z_1, \mathsf{Fin}(\delta_1)) \in \tau, \varrho = \delta_2 = \epsilon$;

- $\mathsf{P}_{\mathsf{fin}}(\cdot) = 1$ if $(z_1, \mathsf{Fail}(\delta_1)) \in \tau, \varrho = \delta_2 = \mathfrak{f}$;

5. the observations $\mathsf{O}_{\mathsf{fin}} = \{Pro(\boldsymbol{B}^f \wedge \boldsymbol{K}\Sigma_\Box, z) | z \in \mathcal{A}^k_{\mathcal{BP}}\}$;
6. the state to observation mapping Ω_{fin} as $\Omega_{\mathsf{fin}}(\langle z, \delta\rangle) = Pro(\boldsymbol{B}^f \wedge \boldsymbol{K}\Sigma_\Box, z)$;
7. the labeling $\mathsf{L}_{\mathsf{fin}}(o) = \{p_\alpha | p_\alpha \in \mathsf{AP}, o \models \alpha\}$.

Obviously, $\mathsf{M}^{e,w}_{\mathsf{Fin},\delta}$ and M^τ_δ have the same state space, action space, and observation space. Additionally, every observation-based policy of $\mathsf{M}^{e,w}_{\mathsf{Fin},\delta}$ is a observation-based policy of M^τ_δ and vice versa. Now we show that they are indeed equivalent:

Proposition 5. *Given a program $\mathcal{BP}$, a bounded state formula Φ', e, w s.t. $e, w \models \Sigma \cup \boldsymbol{B}^f \wedge \boldsymbol{K}\Sigma_\Box$, and a type $\tau = \mathsf{type}(e, w)$, let $\mathsf{M}^{e,w}_{\mathsf{Fin},\delta}$ and M^τ_δ be two POMDPs defined as above. Then for all $z \in \mathcal{A}^k_{\mathcal{BP}}, \delta, \delta' \in \mathsf{Sub}(\delta), \varrho \in \mathsf{Act}, t \in \mathcal{A}_{\mathcal{BP}}$:*

- *for all $\alpha \in \mathcal{C}(\mathcal{BP})$, $e, w, z \models \alpha$ iff $\langle z, \alpha\rangle \in \tau$;*
- $\mathsf{P}^{e,w}(\langle z, \delta\rangle, \varrho, \langle z \cdot t, \delta'\rangle) = \mathsf{P}_{\mathsf{fin}}(\langle z, \delta\rangle, \varrho, \langle z \cdot t, \delta'\rangle)$

where $\mathsf{P}^{e,w}$ is the transition probability of $\mathsf{M}^{e,w}_{\mathsf{Fin},\delta}$.

Proof. First, $e, w, z \models \alpha$ iff $e, w \models [z]\alpha$ iff $\langle z, \alpha\rangle \in \mathsf{type}(e, w)$ (by definition of type).

As for the second, we only consider the case $|z| < k$ and t is a stochastic action, the rest is similar. Suppose $t = a(x, y)$, $y = r_i(x)$, and for some $\phi_j(x)$: $\varrho \to a(x, y), (z, \phi_j(x)) \in \tau$. By the first item, we have $e, w, z \models \phi_j(x)$. Since $e, w \models \Sigma$, therefore $e, w, z \models l(a(x, y)) = r_{i,j}(x)$. On the other hand, $\delta \xrightarrow{\varrho/\alpha} \delta', (z, \alpha) \in \tau, \varrho \to t$ entails $\langle z, \delta\rangle \xrightarrow{e,w} \langle z \cdot t, \delta'\rangle$, by the definition of program characteristic graphs. Hence $\mathsf{P^{e,w}}(\langle z, \delta\rangle, \varrho, \langle z \cdot t, \delta'\rangle) = \mathsf{P}_{\mathsf{fin}}(\langle z, \delta\rangle, \varrho, \langle z \cdot t, \delta'\rangle)$. □

Proof. **of Lemma 1** The lemma is a direct consequence of Prop. 5.

□

References

Bacchus, F., Halpern, J. Y., & Levesque, H. J. (1999). Reasoning about noisy sensors and effectors in the situation calculus. *Artif. Intell.*, *111*(1-2), 171–208.

Baier, C., & Katoen, J. (2008). *Principles of model checking.* MIT Press.

Beauquier, D., Rabinovich, A. M., & Slissenko, A. (2006). A logic of probability with decidable model checking. *J. Log. Comput.*, *16*(4), 461–487.

Belle, V., & Lakemeyer, G. (2017). Reasoning about probabilities in unbounded first-order dynamical domains. In Sierra, C. (Ed.), *Proceedings of the Twenty-Sixth International Joint Conference on Artificial Intelligence, IJCAI 2017, Melbourne, Australia, August 19-25, 2017*, pp. 828–836. ijcai.org.

Belle, V., Lakemeyer, G., & Levesque, H. J. (2016). A first-order logic of probability and only knowing in unbounded domains. In Schuurmans, D., & Wellman, M. P. (Eds.), *Proceedings of the Thirtieth AAAI Conference on Artificial Intelligence, February 12-17, 2016, Phoenix, Arizona, USA*, pp. 893–899. AAAI Press.

Belle, V., & Levesque, H. J. (2013). Reasoning about continuous uncertainty in the situation calculus. In Rossi, F. (Ed.), *IJCAI 2013, Proceedings of the 23rd International Joint Conference on Artificial Intelligence, Beijing, China, August 3-9, 2013*, pp. 732–738. IJCAI/AAAI.

Belle, V., & Levesque, H. J. (2015). ALLEGRO: belief-based programming in stochastic dynamical domains. In Yang, Q., & Wooldridge, M. J. (Eds.), *Proceedings of the Twenty-Fourth International Joint Conference on Artificial Intelligence, IJCAI 2015, Buenos Aires, Argentina, July 25-31, 2015*, pp. 2762–2769. AAAI Press.

Belle, V., & Levesque, H. J. (2018). Reasoning about discrete and continuous noisy sensors and effectors in dynamical systems. *Artif. Intell.*, *262*, 189–221.

Belle, V., & Levesque, H. J. (2020). Regression and progression in stochastic domains. *Artif. Intell.*, *281*, 103247.

Bordini, R. H., Hübner, J. F., & Wooldridge, M. (2007). *Programming multi-agent systems in AgentSpeak using Jason*, Vol. 15. John Wiley & Sons.

Bork, A., Junges, S., Katoen, J., & Quatmann, T. (2020). Verification of indefinite-horizon pomdps. In Hung, D. V., & Sokolsky, O. (Eds.), *Automated Technology for Verification and Analysis - 18th International Symposium, ATVA 2020, Hanoi, Vietnam, October 19-23, 2020, Proceedings*, Vol. 12302 of *Lecture Notes in Computer Science*, pp. 288–304. Springer.

Boutilier, C., Reiter, R., Soutchanski, M., & Thrun, S. (2000). Decision-theoretic, high-level agent programming in the situation calculus. In Kautz, H. A., & Porter, B. W. (Eds.), *Proceedings of the Seventeenth National Conference on Artificial Intelligence and Twelfth Conference on on Innovative Applications of Artificial Intelligence, July 30 - August 3, 2000, Austin, Texas, USA*, pp. 355–362. AAAI Press / The MIT Press.

Chatterjee, K., Chmelik, M., Gupta, R., & Kanodia, A. (2016). Optimal cost almost-sure reachability in pomdps. *Artif. Intell.*, *234*, 26–48.

Chatterjee, K., & Tracol, M. (2012). Decidable problems for probabilistic automata on infinite words. In *Proceedings of the 27th Annual IEEE Symposium on Logic in Computer Science, LICS 2012, Dubrovnik, Croatia, June 25-28, 2012*, pp. 185–194. IEEE Computer Society.

Claßen, J., & Lakemeyer, G. (2008). A logic for non-terminating golog programs. In Brewka, G., & Lang, J. (Eds.), *Principles of Knowledge Representation and Reasoning: Proceedings of the Eleventh International Conference, KR 2008, Sydney, Australia, September 16-19, 2008*, pp. 589–599. AAAI Press.

Claßen, J., Liebenberg, M., Lakemeyer, G., & Zarrieß, B. (2014). Exploring the boundaries of decidable verification of non-terminating golog programs. In Brodley, C. E., & Stone, P. (Eds.), *Proceedings of the Twenty-Eighth AAAI Conference on Artificial Intelligence, July 27 -31, 2014, Québec City, Québec, Canada*, pp. 1012–1019. AAAI Press.

Claßen, J., & Zarrieß, B. (2017). Decidable verification of decision-theoretic golog. In Dixon, C., & Finger, M. (Eds.), *Frontiers of Combining Systems - 11th International*

Symposium, FroCoS 2017, Brasília, Brazil, September 27-29, 2017, Proceedings, Vol. 10483 of *Lecture Notes in Computer Science*, pp. 227–243. Springer.

De Giacomo, G., & Levesque, H. J. (1999). *An Incremental Interpreter for High-Level Programs with Sensing*, pp. 86–102. Springer Berlin Heidelberg, Berlin, Heidelberg.

Fang, L., Liu, Y., & Wen, X. (2015). On the progression of knowledge and belief for nondeterministic actions in the situation calculus. In Yang, Q., & Wooldridge, M. J. (Eds.), *Proceedings of the Twenty-Fourth International Joint Conference on Artificial Intelligence, IJCAI 2015, Buenos Aires, Argentina, July 25-31, 2015*, pp. 2955–2963. AAAI Press.

Feng, Q., Liu, D., Belle, V., & Lakemeyer, G. (2023). A logic of only-believing over arbitrary probability distributions. In Agmon, N., An, B., Ricci, A., & Yeoh, W. (Eds.), *Proceedings of the 2023 International Conference on Autonomous Agents and Multiagent Systems, AAMAS 2023, London, United Kingdom, 29 May 2023 - 2 June 2023*, pp. 355–363. ACM.

Fijalkow, N., Gimbert, H., & Oualhadj, Y. (2012a). Deciding the value 1 problem for probabilistic leaktight automata. In *Proceedings of the 27th Annual IEEE Symposium on Logic in Computer Science, LICS 2012, Dubrovnik, Croatia, June 25-28, 2012*, pp. 295–304. IEEE Computer Society.

Fijalkow, N., Gimbert, H., & Oualhadj, Y. (2012b). Deciding the value 1 problem of probabilistic leaktight automata. *CoRR*, *abs/1104.3055v5*.

Gabaldon, A., & Lakemeyer, G. (2007). ESP: A logic of only-knowing, noisy sensing and acting. In *Proceedings of the Twenty-Second AAAI Conference on Artificial Intelligence, July 22-26, 2007, Vancouver, British Columbia, Canada*, pp. 974–979. AAAI Press.

Giacomo, G. D., Lespérance, Y., & Levesque, H. J. (2000). Congolog, a concurrent programming language based on the situation calculus. *Artif. Intell.*, *121*(1-2), 109–169.

Giacomo, G. D., Lespérance, Y., Patrizi, F., & Sardiña, S. (2016). Verifying congolog programs on bounded situation calculus theories. In Schuurmans, D., & Wellman, M. P. (Eds.), *Proceedings of the Thirtieth AAAI Conference on Artificial Intelligence, February 12-17, 2016, Phoenix, Arizona, USA*, pp. 950–956. AAAI Press.

Giacomo, G. D., Ternovska, E., & Reiter, R. (1997). Non-terminating processes in the situation calculus. In *Proc. of AAAI 97's Workshop on Robots, Softbots, Immobots: Theories of Action, Planning and Control*.

Giacomo, G. D., Ternovska, E., & Reiter, R. (2020). Non-terminating processes in the situation calculus. *Ann. Math. Artif. Intell.*, *88*(5-6), 623–640.

Gimbert, H., & Oualhadj, Y. (2010). Probabilistic automata on finite words: Decidable and undecidable problems. In Abramsky, S., Gavoille, C., Kirchner, C., auf der Heide, F. M., & Spirakis, P. G. (Eds.), *Automata, Languages and Programming, 37th International Colloquium, ICALP 2010, Bordeaux, France, July 6-10, 2010, Proceedings, Part II*, Vol. 6199 of *Lecture Notes in Computer Science*, pp. 527–538. Springer.

Hansson, H., & Jonsson, B. (1994). A logic for reasoning about time and reliability. *Formal Aspects Comput.*, *6*(5), 512–535.

Hardy, G. H., & Wright, E. M. (1995). *An introduction to the theory of numbers (5. ed.)*. Clarendon Press.

Hensel, C., Junges, S., Katoen, J., Quatmann, T., & Volk, M. (2022). The probabilistic model checker Storm. *Int. J. Softw. Tools Technol. Transf.*, *24*(4), 589–610.

Izzo, P., Qu, H., & Veres, S. M. (2016). A stochastically verifiable autonomous control architecture with reasoning. In *55th IEEE Conference on Decision and Control, CDC 2016, Las Vegas, NV, USA, December 12-14, 2016*, pp. 4985–4991. IEEE.

Kemeny, J. G., Snell, J. L., & Knapp, A. W. (2012). *Denumerable Markov chains: with a chapter of Markov random fields by David Griffeath*, Vol. 40. Springer Science & Business Media.

Kwiatkowska, M. Z., Norman, G., & Parker, D. (2009). Stochastic games for verification of probabilistic timed automata. In Ouaknine, J., & Vaandrager, F. W. (Eds.), *Formal Modeling and Analysis of Timed Systems, 7th International Conference, FORMATS 2009, Budapest, Hungary, September 14-16, 2009. Proceedings*, Vol. 5813 of *Lecture Notes in Computer Science*, pp. 212–227. Springer.

Kwiatkowska, M. Z., Norman, G., & Parker, D. (2011). PRISM 4.0: Verification of probabilistic real-time systems. In Gopalakrishnan, G., & Qadeer, S. (Eds.), *Computer Aided Verification - 23rd International Conference, CAV 2011, Snowbird, UT, USA, July 14-20, 2011. Proceedings*, Vol. 6806 of *Lecture Notes in Computer Science*, pp. 585–591. Springer.

Lakemeyer, G., & Levesque, H. J. (2011). A semantic characterization of a useful fragment of the situation calculus with knowledge. *Artif. Intell.*, *175*(1), 142–164.

Lang, J., & Zanuttini, B. (2015). Probabilistic knowledge-based programs. In Yang, Q., & Wooldridge, M. J. (Eds.), *Proceedings of the Twenty-Fourth International Joint Conference on Artificial Intelligence, IJCAI 2015, Buenos Aires, Argentina, July 25-31, 2015*, pp. 1594–1600. AAAI Press.

Levesque, H. J., & Lakemeyer, G. (2001). *The logic of knowledge bases*. MIT Press.

Levesque, H. J., Reiter, R., Lespérance, Y., Lin, F., & Scherl, R. B. (1997). GOLOG: A logic programming language for dynamic domains. *J. Log. Program.*, *31*(1-3), 59–83.

Lin, F. (2016). A formalization of programs in first-order logic with a discrete linear order. *Artif. Intell.*, *235*, 1–25.

Lin, F., & Reiter, R. (1997). How to progress a database. *Artif. Intell.*, *92*(1-2), 131–167.

Liu, D., & Feng, Q. (2023). On the progression of belief. *Artif. Intell.*, *322*, 103947.

Liu, D., Feng, Q., Belle, V., & Lakemeyer, G. (2023a). Concerning measures in a first-order logic with actions and meta-beliefs. In Marquis, P., Son, T. C., & Kern-Isberner, G. (Eds.), *Proceedings of the 20th International Conference on Principles of Knowledge Representation and Reasoning, KR 2023, Rhodes, Greece, September 2-8, 2023*, pp. 451–460.

Liu, D., Huang, Q., Belle, V., & Lakemeyer, G. (2023b). Verifying belief-based programs via symbolic dynamic programming. In Gal, K., Nowé, A., Nalepa, G. J., Fairstein, R.,

& Radulescu, R. (Eds.), *ECAI 2023 - 26th European Conference on Artificial Intelligence, September 30 - October 4, 2023, Kraków, Poland - Including 12th Conference on Prestigious Applications of Intelligent Systems (PAIS 2023)*, Vol. 372 of *Frontiers in Artificial Intelligence and Applications*, pp. 1497–1504. IOS Press.

Liu, D., & Lakemeyer, G. (2021). Reasoning about beliefs and meta-beliefs by regression in an expressive probabilistic action logic. In Zhou, Z. (Ed.), *Proceedings of the Thirtieth International Joint Conference on Artificial Intelligence, IJCAI 2021, Virtual Event / Montreal, Canada, 19-27 August 2021*, pp. 1951–1958. ijcai.org.

Liu, Y., & Levesque, H. J. (2005). Tractable reasoning with incomplete first-order knowledge in dynamic systems with context-dependent actions. In Kaelbling, L. P., & Saffiotti, A. (Eds.), *IJCAI-05, Proceedings of the Nineteenth International Joint Conference on Artificial Intelligence, Edinburgh, Scotland, UK, July 30 - August 5, 2005*, pp. 522–527. Professional Book Center.

Luckcuck, M., Farrell, M., Dennis, L. A., Dixon, C., & Fisher, M. (2019). Formal specification and verification of autonomous robotic systems: A survey. *ACM Comput. Surv.*, *52*(5), 100:1–100:41.

Macintyre, A., Wilkie, A. J., & Odifreddi, P. (1996). On the decidability of the real exponential field. *Kreisel's Mathematics*, *115*, 451.

Madani, O., Hanks, S., & Condon, A. (2003). On the undecidability of probabilistic planning and related stochastic optimization problems. *Artif. Intell.*, *147*(1-2), 5–34.

McCarthy, J. (1963). Situations, actions, and causal laws. Tech. rep., Stanford University.

McCarthy, J., & Hayes, P. J. (1981). Some philosophical problems from the standpoint of artificial intelligence. In *Readings in artificial intelligence*, pp. 431–450. Elsevier.

Norman, G., Parker, D., & Zou, X. (2017). Verification and control of partially observable probabilistic systems. *Real Time Syst.*, *53*(3), 354–402.

Paz, A. (2014). *Introduction to probabilistic automata*. Academic Press.

Reiter, R. (2001). *Knowledge in action: Logical foundations for specifying and implementing dynamical systems*. MIT Press.

Sanner, S., & Boutilier, C. (2009). Practical solution techniques for first-order mdps. *Artif. Intell.*, *173*(5-6), 748–788.

Sanner, S., & Kersting, K. (2010). Symbolic dynamic programming for first-order pomdps. In Fox, M., & Poole, D. (Eds.), *Proceedings of the Twenty-Fourth AAAI Conference on Artificial Intelligence, AAAI 2010, Atlanta, Georgia, USA, July 11-15, 2010*, pp. 1140–1146. AAAI Press.

Sardiña, S., Giacomo, G. D., Lespérance, Y., & Levesque, H. J. (2004). On the semantics of deliberation in indigolog - from theory to implementation. *Ann. Math. Artif. Intell.*, *41*(2-4), 259–299.

Scherl, R. B., & Levesque, H. J. (2003). Knowledge, action, and the frame problem. *Artif. Intell.*, *144*(1-2), 1–39.

Soutchanski, M. (2001). An on-line decision-theoretic golog interpreter. In Nebel, B. (Ed.), *Proceedings of the Seventeenth International Joint Conference on Artificial Intelligence, IJCAI 2001, Seattle, Washington, USA, August 4-10, 2001*, pp. 19–26. Morgan Kaufmann.

Specker, E. (1949). Nicht konstruktiv beweisbare sätze der analysis. *The Journal of Symbolic Logic*, *14*(3), 145–158.

Tarski, A. (1998). A decision method for elementary algebra and geometry. In *Quantifier Elimination and Cylindrical Algebraic Decomposition*.

Vassos, S., & Levesque, H. J. (2008). On the progression of situation calculus basic action theories: Resolving a 10-year-old conjecture. In Fox, D., & Gomes, C. P. (Eds.), *Proceedings of the Twenty-Third AAAI Conference on Artificial Intelligence, AAAI 2008, Chicago, Illinois, USA, July 13-17, 2008*, pp. 1004–1009. AAAI Press.

Zarrieß, B., & Claßen, J. (2014). Verifying ctl* properties of GOLOG programs over local-effect actions. In Schaub, T., Friedrich, G., & O'Sullivan, B. (Eds.), *ECAI 2014 - 21st European Conference on Artificial Intelligence, 18-22 August 2014, Prague, Czech Republic - Including Prestigious Applications of Intelligent Systems (PAIS 2014)*, Vol. 263 of *Frontiers in Artificial Intelligence and Applications*, pp. 939–944. IOS Press.

Zarrieß, B., & Claßen, J. (2015). Decidable verification of knowledge-based programs over description logic actions with sensing. In Calvanese, D., & Konev, B. (Eds.), *Proceedings of the 28th International Workshop on Description Logics, Athens,Greece, June 7-10, 2015*, Vol. 1350 of *CEUR Workshop Proceedings*. CEUR-WS.org.

Zarrieß, B., & Claßen, J. (2016). Decidable verification of golog programs over non-local effect actions. In Schuurmans, D., & Wellman, M. P. (Eds.), *Proceedings of the Thirtieth AAAI Conference on Artificial Intelligence, February 12-17, 2016, Phoenix, Arizona, USA*, pp. 1109–1115. AAAI Press.